Command Line: First tour

- Today's Slideshow

Our main goal is to set up everyone clients, and to get the main tools to work in the future sessions: ability to connect to the remote server, using `screen` to keep a living session on the remote server.

General rules

- ❗ Read carefully each paragraph before typing commands.
- ❗ Use the **TAB completion**. Always.
- ❗ Strictly follow the instructions using the suggeste filenames: creativity is not a positive quality for Bash beginners ;)

Accessing your remote machine

Follow instructors instructions to log in into your machine. Generally speaking you need to know:

- The address of the remote machine (can be an IP like *44.20.200.12* or a string like *server4* or *nasa.org*)
- Your username
- Your password

Review the page on remote access for further details.

Where are you?

By default, when you log into a remote machine, you'll find yourself welcomed in your **home directory**. This is a special location dedicated to you, where you can save some files. On a typical set up its path will be `/home/username/`, but on complex servers with many users this can vary widely. To check your location, use the pwd (print working directory) command:

```
pwd
```

Remember that the output will be the *absolute path* of your current location.

While it might sound strange to have identity problems at your young age ;), if you use multiple machines it could be that you have more usernames. So if you are unsure about who you are (*i.e.* what is your current active username):

```
whoami
```

What's in your current directory? Try the `ls` (list) command to see:

```
ls
```

We can also add a first *switch* to this command: to view more details use the `-l` (*long*) switch:

```
ls -l
```

The shell prompt

Now that you tried the first commands, let's make a step back. The linux shell has a string to tell you it's ready to receive commands. The string is called **shell prompt**, and can be very minimal, like a simple $, or a more complex thing. As you can guess, it's something we can customize. Usually has a structure like:

```
telatin@gmh:~$
```

That is: *username* @ *machine name* : *current directory* $

The last character is conventionally a dollar sign ($) for regular users, and a sharp (#) for super-user (administrators). In Linux there is only one administrator called *root*, but some users could have the privilege to impersonate *root* from time to time.

Also note that the tilda (~) is a shortcut for your home directory.

Remember that you have to wait to see the shell prompt before issuing new commands. If you don't see it, maybe the previous command is stuck or simply still working.

Using a demo "program"

Type the command:

```
weather.pl
```

Many programs will give you a short help message using the switches `-h` or `–help`. Try with one of these, then get the weather for Cambridge.

Using screen

Before moving on with the workshop, we will start using a powerful program for a safer and better work on a remote server: **screen**.

Assuming you never used screen on the server, you can create a new session using the command:

```
screen -S course
```

More details on this: Using screen tutorial or view a video on this

Navigating the filesystem

Now try changing directory, that means setting a new working directory:

```
cd /tmp
```

Did it work? Test with pwd.

To return quickly to the home directory, you can use **cd** without specifying the destination path:

```
cd
```

Now we are back to our home directory. So we can create a new sub-directory called *course*:

```
mkdir course
```

More details on this, but we'll make more examples on Day2: Introduction to the filesystem

Using the manual for a command

Bundled with your distribution there is a manual. It's useful to know that is there, but it's even more useful understanding how to use it as we'll see that another very handy shell command has the very same behaviour.

The manual command is `man`, followed by the command:

```
man ls
```

When launching the manual, you'll enter an interactive page. You can:

- Scroll using the arrow keys, or "Page Up" and "Page down", or the space bar to scroll down one page
- g will go quickly to the beginning of the document, G to the end
- / enable a serch inside the document: type "/", a string and then ENTER
- After a search n will jump to the next occurrence, N to the previous
- Finally, q to quit

Text editor: nano

❗ Try this part yourself when you have five minutes spare.

To create or edit a text file directly from the shell, there is a small editor called **nano**. Its basic usage is

```
nano filename.txt
```

It works like any text editor, and at the bottom there is an array of shortcuts (^ means Ctrl). For example `Ctrl+X` to exit, or `Ctrl+W` to search for a word.

This small video shows how to use it: