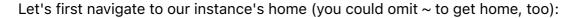
Bash for HPC -- WORK IN PROGRESS

Handling directories and files



\$ cd ~

Check the path of the current working directory:

\$ pwd

/home/ubuntu

Go to the parent directory of the current one:

\$ cd ..

Check where you are:

\$ pwd

/home

Now navigate to the data directory for this workshop, /data:

\$ cd /data

To make a new file, e.g. an empty text file:

\$ touch empty.txt

We can check the content of this directory, to see whether the file has been created:

\$ 1s

empty.txt

Get details about files and directories with:

\$ ls -l

```
-rw-r---- 1 ubuntu ubuntu 0 Jun 6 15:24 empty.txt
```

To make a copy of a file with another name:

```
$ cp empty.txt empty-copy.txt
```

To rename a file:

```
$ mv empty-copy.txt empty-second.txt
```

To move a file to another directory, e.g. your home (still mv, but now the second argument is an existing directory):

```
$ mv empty-second.txt /home/ubuntu/
```

Now let's remove both files:

```
$ rm empty.txt /home/ubuntu/empty-second.txt
```

To discover the location of an application that is available in the environment:

```
$ which bash
/bin/bash
```

Defining variables

We can define a variable for a string of characters, e.g. a name, a filename or a directory. The variable can be used later on in the script or in the shell to refer to that string:

```
$ HELLO="ciao"
```

We can check that it was defined correctly:

```
$ echo $HELLO
ciao
```

If we want the variable to be accessible inside sub-processes, we need to export it:

```
$ export WORLD="mondo"
```

A process could be a container, or just another bash shell:

```
$ bash -c 'echo $HELLO $WORLD'
mondo
```

Only the exported one is available in the sub-process.

We might use a variable for a directory path:

```
$ MYHOME="/home/ubuntu"
```

Now we can use this variable with commands that operate on directories:

```
$ cd $MYHOME
$ pwd
/home/ubuntu
```

There's a way to store the output of a command inside the variable. It uses the form \$(..). For instance, let's obtain the path of the current working directory from the command cmd, and store it in the variable myplace:

```
$ myplace=$(pwd)
$ echo $myplace
/home/ubuntu
```

Every shell session automatically defines a number of variables, some of which can be useful.

Among others, \$HOME contains the path to your home directory, \$USER contains your username, and \$PATH has the list of paths where the shell looks for executable programs and scripts.

Visualising contents of text files

Let's now create a text file with some words in it. We'll use a popular, friendly text editor called nano:

```
$ nano words.txt
```

In the editor screen, write something random.

To save this text, press Ctrl-0, then Enter. To exit, press Ctrl-X.

Now we want to visualise the contents of this file on our terminal.

We can use cat to display it all in one go:

\$ cat words.txt
random words that we just saved in this file

For very long files, we might want to be able to navigate the content in chunks. We can achieve this with less:

\$ less words.txt

Go forward by pressing Ctrl-F, go backwards with Ctrl-B. Quit the navigation by pressing q.