

Tutorial 1

Bash fundamentals

`cat`, displays the output of a file.

```
$ cat test.txt
This is the content of test.txt
```

`cd`, change directory, for example if your current directory contains a folder named `folderB` you can go to `folderB` by using the `cd` command:

```
$ cd folderB/
```

If you want to go up one directory, you can use:

```
$ cd ../
```

`chmod`, change permission of a file. If you create a file, you can prevent other users from reading writing or executing this file:

```
$ chmod 000 test.py # withdraw permission to read, write and execute

$ cat test.py # try to read the file
cat: test.txt: Permission denied

$ echo "test" > test.py # try to write the file
-bash: test.py: Permission denied

$ python test.py # try to execute the file
python: can't open file 'test.py': [Errno 13] Permission denied
```

You can give permission to read, write and execute:

```
$ chmod 777 test.py # permission to read, write and execute

$ cat test.py # now you can read the file
```

```
print ('hello world')

$ echo "print ('hello')" > test.py # write the file

$ python test.py # and execute the file
hello
```

In practise, when you upload a script to a server, you can not execute it by default (for security reasons), so you have to use the `chmod` to grant permission to execute script.

`cp`, copy a file to a given directory, for example if you have a file named `test.txt` in your current directory and you want to copy this file to your desktop, you can use:

```
$ cp test.txt ~/Desktop
```

`echo`, return a given value, since the functionality of the `echo` commands can vary, you can also use `printf` (same functionality but more concise).

```
$ echo "Hello"
Hello
```

`grep`, filters a given input. This command is often used in combination with the `history` command, to search through all executed commands.

```
$ history | grep chmod
[...]
508  chmod 000 test.py
510  chmod 777 test.py
520  history | grep chmod
```

`history`, returns a list of executed command, this comes in handy if you used a command in and can't remember it. See also `grep` command.

`mkdir`, creates a directory. For example if you want to create a folder named `folderA`, you can use the following command:

```
$ mkdir folderA
```

mv, move a file to another directory, for example if you have a `test.txt` file in your current directory and want to copy this file to your desktop, you can use:

```
$ mv test.txt ~/Desktop
```

ls, returns the names of the files and directories in you current directory. You can also display addition information (permissions, owner, date of change) by using the `-l` flag:

```
$ ls
test.py      test.txt

$ ls -l
total 16
-rwxrwxrwx  1 laurens  staff  16 Jul 13 09:55 test.py
-rw-r--r--  1 laurens  staff   7 Jul 13 09:55 test.txt
```

paste, merge two output streams by column, for example if you have two files and want to create a new file with the content of both files as two columns:

```
$ paste file1.txt file2.txt > filerresults.txt
```

pwd, returns the full path of you current directory:

```
$ pwd
/Users/laurens/Desktop/folderA
```

rm, removes a file:

```
$ rm test.txt
```

Or an entire directory (including all content):

```
$ rm -r folderA
```

`sort`, is used to sort an input (comparable with the filter function of `grep`). Use the `-r` flag to reverse the sort and use `-n` for numeric sort.

```
$ echo -e "B\nA\nC" | sort # sort in alphabetical order
```

```
A
```

```
B
```

```
C
```

```
$ echo -e "B\nA\nC" | sort -r # reverse sort
```

```
C
```

```
B
```

```
A
```

```
$ echo -e "100\n2100\n30" | sort # alphabetical order with number
```

```
100
```

```
2100
```

```
30
```

```
# echo -e "100\n2100\n30" | sort -n # numerical order
```

```
30
```

```
100
```

```
2100
```

Common symbols

| Symbol | Meaning | Example |
|--------|---|-------------------------|
| | Get the output of one command and use it as input from the next command | history grep ls |
| > | Get the output of a command and write it to a file | echo "test" > test.txt |
| >> | Get the output of a command and add it to a file | echo "test" >> test.txt |
| * | Placeholder for every character or sequence | find *.txt |

| | | |
|------------|---|---|
| ~ | Home directory | <code>cd ~/</code> |
| # | Indicate comment (everything on the same line after # is ignored) | <code>cd ~/ # go to home directory</code> |
| .. | Parent directory | <code>cd ../ # go one directory up</code> |
| <TAB> | Not really a symbol but the tab-key is useful for autocompletion | <code>his<TAB><TAB> # results in history</code> |
| <ARROW UP> | The arrow-up-key is useful for getting the previous command. | |
| \n | Character for a new line | <code>echo -e "line 1\nline 2"</code> |
| \t | Character for a tab | <code>echo -e "col 1\tcol 2"</code> |

Shortcuts

| | | |
|--------------|----------------------------------|--|
| Ctrl+c | Kill a running program. | For example, the <code>ping</code> command run infinitely so you can stop by using <code>Ctrl+c</code> |
| Shift+Ctrl+c | Copy text from the command line. | Select interesting output from the command line and use <code>Shift+Ctrl+c</code> to copy it to your clipboard (on linux). |
| Shift+Ctrl+v | Paste text in you command line | Select an interesting command and copy it with <code>Ctrl+c</code> (as usual) and paste it in your command line with <code>Shift+Ctrl+v</code> (on linux). |

Creating bash scripts

Commands are useful, but in practise it often takes a lot of time to type and retype every command. For this reason you can create a bash-script to bundle a number of commands. Creating a script is easy (just create a text-file with a `.sh` extension) then you need to give this file permission to execute and then you can run it, for example:

```
$ echo "echo hello" > hello.sh # create a script to display hello
$ chmod 777 hello.sh # grant permission to execute
```

```
$ ./hello.sh # execute the script
hello
```

A more useful example is `dss_grade_calc.sh` which can calculate your final grade for the DSS course:

```
#!/bin/bash

# getting the grades from the argument
book=$1
mid_term=$2
end_term=$3

# calculate your final grade using bc (basic calculator)
final_grade=$(expr $book*.10+$mid_term*.40+$end_term*.50 | bc)

# show the result
echo "your final grade is: $final_grade"
```

To run this script, you have to create a text file with the content as shown above (for example by using `nano`), grant permission to execute this file and execute it (providing your grades as arguments):

```
$ nano dss_grade_calc.sh # then do some copy-pasting and save
$ chmod 777 dss_grade_calc.sh # Grant permission to execute
$ ./dss_grade_calc.sh 8 6.5 7 # Arguments: book, mid term, end term
Your final grade is: 6.90
```

Assignments

0.0. Please change your password if have not done this already!

Since the virtual environments are an easy target for hackers, it is highly advised to change the password of your virtual environment. You can change your password by using the following commands:

```
$ sudo passwd labuser # Where labuser is your username.
```

After this command you can choose a new password, there is no feedback when you type your password. After your done press enter to apply the new password.

1.1. Navigate to your home directory and create a folder named `tutorial_1`, go to this folder and create the following structure (where `folderA` contains `folderB`)

```
tutorial_1
  folderA
    folderB
  folderC
  folderD
```

1.2. Create a text file in `folderB`, named `file_0.txt`, which contains the following five lines:

```
row 1
row 2
row 3
row 4
row 5
```

1.3. Create a text file in `folderB` named `file_r.txt`, with the same lines as `file_0.txt` but in reverse order.

1.4. Move `file_0.txt` to `folderA` and make a copy of `file_r.txt` and move that copy to `folderC`.

1.5. Make a new file in `folderD`, named `file_1.txt` with the content of `file_0.txt` and append the following lines:

```
row 6
row 7
row 8
row 9
row 10
```

1.6. Go one directory up and remove `folderD`.

1.7. Merge `file_0.txt` and `file_r.txt` into a new file named `file_m.txt` so that `file_m.txt` looks like:

```
row 1      row 5
row 2      row 4
row 3      row 3
row 4      row 2
row 5      row 1
```

1.8. Use the `tree` command to view your folder structure, which should look like:

```
.
|___file_m.txt
|___folderA
| |___file_0.txt
| |___folderB
| | |___file_r.txt
|___folderC
| |___file_r.txt
```

Question for Honor students: extent the `dss_grade_calc.sh` script to indicate if a student passed the course (e.g. if the final grade is 5.5 or above echo a “congratulations”). If you want to apply for a honor notification, you can submit this question to marco with subject HONOR INFOMDSS.

At the end of the workshop stop your virtual machine