

# Lab4

## The environmental variable PATH. First Bash Scripts.

Read carefully the instructions before performing!!!

Submission in pairs : 09/12/24 for the groups of Monday  
12/12/24 for the groups of Thursday

- Standard commands using by PATH.
- Chmod command
- Let's write our first script in bash!
- Variables in bash
- Command substitution
- Arithmetics in Bash

### 1. Linux Fundamental directories

~ = Your home directory

/ = Root directory (Root is all powerful. Root access is restricted to system administrator)

/bin = binaries and other executable programmes

/etc = system configuration files

/home = home directories

/usr = user related programs

/var = variable data

/usr/local = User applications can be installed in usr/local or /apt

### 2. The environmental variable PATH

The environmental variable PATH controls the command search path.

> **echo \$PATH** = shows list of directories in which there are standard programs.

When we use, for example, the command **cat**, the system is looking for the program named cat in all the directories, defined in PATH.

# Linux Commands

1. **which** file\_name = find the location of the file\_name

Example:

```
> which cat
```

```
/usr/bin/cat
```

Actually, when we write

```
> cat foo.dat
```

the program named cat which is located in the path /usr/bin/cat is running.  
Other way to run it is simply to write the full path of the running program:

```
>/usr/bin/cat foo.dat
```

**2.chmod** - change permissions of the files

**EXAMPLE**

```
> ls -l foo
```

```
drwxrwxrwx  1 root root 505 Nov 04 22:30 foo
```

**d** = type (**d** for directories and - for files)

**d**            **rw****x**                            **rw****x**                            **rw****x**

(Type)    (User permissions) (Group permissions) (Others permissions)

r = permission to read the file

w = permission to write to the file

x = permission to execute the file

- = no permission to read/write/execute

**EXAMPLE**

```
-rw-r--r-- 1 root root  46 Apr 14 16:37 example
```

**File,**

**User:** can read and write

**Group:** can read only

## Others: can read only

---

\* ugoa - user/group/other/all

\*+ -= - Add/Sub/Equal

\* rwx - read/write/execute

### EXAMPLES

>chmod g+rwx example (**Add** the permissions for the group to be rwx)

>chmod g-r example (Remove the permission to read for the group)

>chmod u+rwx , g-w example (Add rwx to user and remove w for group)

>chmod **a=r** example (Permission only to read for **all**)

### Numerically based permissions

r	w	x	
0	0	0	Value for OFF
1	1	1	Binary ON
4	2	1	ON Based on 10

Octal	Binary	String
0	000	---
1	001	--X
2	010	-W-
3	011	-WX
4	100	r--
5	101	r-X
6	110	rw-
7	111	rwx

### EXAMPLES

>chmod +700 example (Means -rwx-----)

>chmod +755 example (Means -rwxr-xr-x)

# ShortCuts

## 1. Reusing the previous commands.

To reuse the command which was used before, press few times the button



## 2. Completion of commands using arrows.

Write the command and then the name of the file and then press.



The filename will be completed, if possible.

# Bash

1. Each file is in the format name.**sh**

2. The file starts with **#!/bin/bash**.

A shebang (#!) is a special line at the beginning of a script that tells the operating system which interpreter to use when executing the script. This line is the first line of a script and starts with "#!" followed by the path to the interpreter.

3. Variables in bash can be defined by **VAR="val"**, where val is a string.

**Attention:** No spaces here!! The name cannot start with a number!!

Examples:


MY\_VAR="I write Bash"

MY1Var = "2"

my123Var="2+3"

Invalid name: 123myVAR="2+3"

4. Command substitution can be done using

\$(command) or \$(command1|command2|command3) or `command` or `command1|command2|command3`, where the button  can be found above the Tab button.

5. Arithmetics can be done using **\$((name1 op name 2))**.

Four ops are possible *in integers*: +, -, \*, /.

6. To read input from the STDIN, use the command

read -p "Text\_asking\_input" NAME\_OF\_VAR

## Example 1 of file MyFirstBash.sh

```
#!/bin/bash
```

```
VAR1="boo likes docu"
```

```
VAR2="2"
```

```
VAR3="3"
```

```
VAR4="drama"
```

```
echo "VAR1=$VAR1"
```

```
echo "VAR2=$VAR2, VAR3=$VAR3"
```

```
echo "and together it means $VAR1$VAR4"
```

```
echo "$VAR2+$VAR3 = $(( $VAR2+$VAR3 ))"
```

Running:

```
> chmod 777 MyFirstBash.sh
```

```
> ~/Lab4/MyFirstBash.sh
```

### **Example 2 of file MySecondBash.sh**

```
#!/bin/bash
echo "$ (cat MyFirstBash.sh | tail+3 |head -2)"
echo "my present working directory is `pwd`"
echo "the directory $(pwd) contains the following files: $(ls)"
```

#### Running:

```
>chmod +755 MySecondBash.sh
> ./MySecondBash
```

### **Example 3 of file ReadInput.sh**

```
#!/bin/bash
read -p "Enter your input:" INPUT_VAR
echo "your input values = $INPUT_VAR"
read -p "Enter 3 input values:" VAR1 VAR2 VAR3
echo "your input values = $VAR1 $VAR2 $VAR3"
echo "the sum of these values is $(( $VAR1+$VAR2+$VAR3 ))"
```

## **Procedures**

1. Print the word PATH.
  2. Print the value of the env. variable PATH.
  3. Find in which directory the standard commands touch, ls and find are defined.
  4. Create a new directory Lab4.
  5. In Lab4 create 10 files: - 4 files which contain the letter m in their names  
- 6 files which contain a number in their names
  6. In Lab4 create a new directory SubLab4Dir.
  7. In SubLab4Dir create 5 files: - 1 file which contain the letter t in the name  
- 3 files which contain a number in the name
  8. In Lab4 write bash script named **FirstBash.sh** which is doing the following:
    - Print the word PATH.
    - Print the value of the env. variable PATH.
    - Find in which directory the standard commands touch, ls and find are defined.
- Run this script and check that the results are correct.

9. In Lab4 write bash script named **FindFiles.sh** which is doing the following:

- show the contain of the directory Lab4.
- print the list of the files located in Lab4 which contain the letter **m** in their names.
- go to the subdirectory SubLab4Dir.
- show the contain of the subdirectory SubLab4Dir.
- print list of the files located in SubLab4Dir which contain a number in their names.
- print the name of the file located in SubLab4Dir which contain a letter **t** in its name. The name should contain the full-path name (for example ~/home/moshe/SubLab4Dir/klk1112klk.out)

**Note:** Assume that you know the name of the file with a letter t. You should print only one specific name of the file created by you above. *(You don't need to use any loop here! Use only the commands taught in the course.)*

Run this script and check that the results are correct.

10. In Lab4 write bash script named **MathBash.sh**

- Ask 4 numbers in range (1-8) from the input (there is no need to check the correctness of the input).
- Print the sum and the product of these numbers.
- Rename the file of directory SubLab4Dir, which contains the letter **t** so the last letter of it will be one of the numbers you received from the input.

For example if one of the input numbers was 2 and if SubLab4Dir contains file Mat.n, then after running of MathBash.sh, SubLab4Dir will contain (instead of this file), file named Mat.n2.

**Note:** Assume that you know the name of the file with a letter t. You should print only one specific name of the file created by you above. *(You don't need to use any loop here! Use only the commands taught in the course.)*

*in the course)*

## Submission:

Create one compressed file named **Linux\_Lab4\_id1\_id2.tar**, where id1 and id2 must be changed to the id numbers of the two partners.

This file will include the files FirstBash.sh, FindFiles.sh, MathBash.sh.

To create such file in Linux command line write:

```
$ tar -cvf Linux_Lab4_id1_id2.tar FirstBash.sh FindFiles.sh MathBash.sh
```

Submit **only** the file **Linux\_Lab4\_id1\_id2.tar** in Moodle and only by **one** of the partners.

## **Appeals:**

Can be submitted by e-mail to Elad [mail@eladhuttner.net](mailto:mail@eladhuttner.net) within a week from the date of publication of the grades.