Lab 7
Cut command.
Positional Parameters.
Functions in Bash.

Read carefully the instructions before performing!!!
Submission in pairs : 06/01/25 for the groups of Monday
09/01/25 for the groups of Thursday

Linux Commands

cut = cuts parts of a line by byte position, character, and field.

- cut OPTIONS file

Options:

-c : cuts characters

-f : cuts fields

-d: specify the field separator

Notes:

1). If the delimiter is present, but the number of fields in the line is less than the requested field number, an empty line will be printed.
2). If there is no separator character in the line, the whole line will be printed.

Example

>cat **cutFile**

a11 a12 a13 a14 a15

b21 b22 b23 b24 b25

c31 c32 c33 c34 c35

>cut **-c1-3,5,8-10** cutFile

a11a a1

b21b b2

c31c c3

>cut **-d" " -f2,4** cutFile

a12 a14

b22 b24

c32 c34

>cat **cutFile2**

a11:a12:a13

b21:b22

c31 c32 c33

>cat cutFile2|**cut -d":" -f1**

a11

b21

c31 c32 c33

>cat cutFile2|**cut -d":" -f3**

a13

c31 c32 c33

Bash

1.Positional Parameters

When we ran some command of program by

> program_name word1 word2 word3

we can read each item on the command line because the positional parameters contain the following:

- $0 contains "program_name"
- $1 contains "word1"
- $2 contain "word2"
- $3 contains "word3"
- $# contains number of parameters (not incliding the program_name)
- $@ contains all the parameters (not incliding the program_name)

Example PosPar.sh:

```
#!/bin/bash
#WAY1
echo "Positional Parameters"
echo The program name  '$0 = ' $0
echo The first parameter  '$1 = ' $1
echo The second parameter '$2 = ' $2
echo The third parameter'$3 = ' $3
echo The total number of parameters is $#

#########################################################
#WAY2
i=1
for  PARAM in $@
do
	echo "The parameter number $i is $PARAM"
	i=$((i+1))
done

#########################################################
#WAY3

NUM_OF_PARAM=$#
for (( i=1; i<=NUM_OF_PARAM; i++ ))
do
	echo "name=$0"
	echo "The parameter number $i is $1 "
```

```
        shift
done
```

Note: shift command shifts all the arguments to the left.

Runing:

./PosPar.sh fir sec third

2.Functions

Defining function:

```
function func_name () {
        commands
}
```

```
func_name () {
        commands
}
```

Calling function:

func_name

Local Variables:

Can be defined in the function by:

local var_name

Input Parameters

Can be passed using **positional parameters**

Return value

Can be passed using **command substitution**

Notes:

- The **let** command performs various arithmetic, bitwise and logical operations. The built-in command works only with integers.

- Useful math operators: +, -, *, /, ++, --, +=, -=.

Example FuncExample.sh:

```
#!/bin/bash
##########################################
#Call function
function hello1 () {
	echo "Hello!"
}
hello1  #Call function
##########################################
# Function with one input parameter
function hello2 () {
	echo "Hello $1"
}
hello2 Tomer  #Call function with parameter: Tomer is $1 parameter of hello2
		#Output is Hello Tomer

##########################################
# Function with number of input parameters
hello3 () {
for NAME in $@
do
	echo "Hello $NAME"
done
}

hello3 Miri Natali Yasmin
```

```
##########################################
# Function with output value
sum () {
result=0
for NUM in $@
do
      let result+=$NUM       # Also possible result=$((result+NUM))
done

echo $result
}

n=`sum 1 2 3`  # Run sum function with 3 specific input parameters
echo "n=$n"

n=`sum $@`   #Run sum function with input parameters of FuncExample.sh
echo "new n=$n"

##########################################
# Function with local parameters
surprise () {
local a=surprise_a
b=surprise_b
}
a=original_a
b=original_b
echo $a $b
surprise
echo $a $b
```

Procedures

1.Create directory Lab7.

2. In file **MathFunc.sh,**

a). write a function called **mult** which multiplies together a sequence of numbers.

For example, **mult 2 4 6** will print **48.**

b). Write a function called isEven that prints 1, if a number is even or 0, if a number is not even.

For example, **isEven 24** will print **1**.

c). Write a function called numOfEvens which prints the number of even numbers when provided with a sequence of numbers. Use isEven when writing this function.

For example, numOfEvens **2 5 7 5 8 93** will print **2**.

d). Write a function called **fibo** which prints the number of fibonaci numbers specified.

For example, **fibo 4** will print **0 1 1 2**

and **fibo 10** will print **0 1 1 2 3 5 8 13 21 34**

e). Write a function called minMax which prints the maximum and minimum values of a sequence of numbers.

For example, **minMax 2 67 4 21 9 100 4** will print min=2, max=100.

f). Write a script that takes sequence of numbers, asks the user to choose one of the above functions a), c) or e), then calls the chosen function with this sequence of numbers and prints the result on the screen. If a wrong option is selected, the program will send an appropriate message to the **Errors.er** file.

For example, running the command

> ./ **MathFunc.sh 4 1 2 1**

and choosing the function a), will print:

The result of mult is 8.

3). In Lab7 create a file named **ComposersFile** which contains the First names, Second names, years of birth, countries of birth of at least 5 famous composers. The fields will be separated by spaces

For example, one line of this file can be:

Wolfgang Mozart 1756 Austria.

a). Write a function **first_names** which will create file named FirstYear with First Names and years of birth only of the composers. This functions should be written in the script **Composers.sh** (see the b) part below).

b). Write a script **Composers.sh** which will takes as input one parameter (FN, SN, Year, Country) and prepare file which name is identical to the given parameter and it contains only the information about the First Name/Second name/Year of Birth/Country of birth.

For example,

- **Composers.sh FN** will prepare file named FN which contains the first names of the composers only.

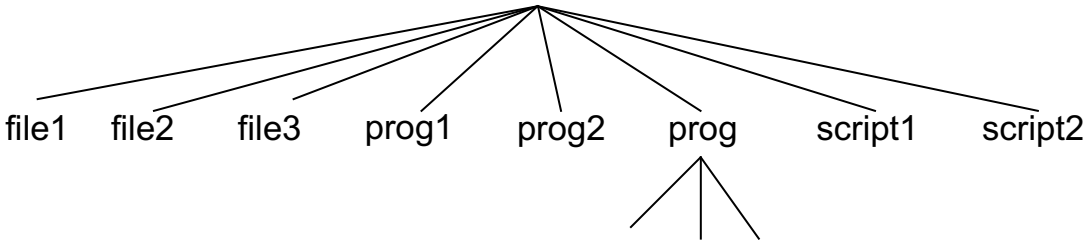
- **Composers.sh Year** will prepare file named Year which contains the years of the birth of the composers only.

4). Write a script **Arrange.sh** that transfers files (which are not directories) from the current directory into sub-directories according to the following rules.

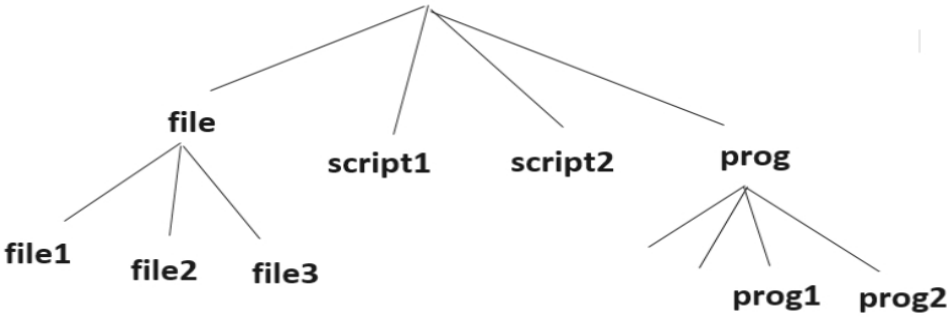
For any file (which is not directory):
1. If there is a sub-directory of the current directory whose name is the same as the first 4 letters in the name of the file, the file will be moved into the directory. (Note that this moves the file and not copies it).

2. If condition 1 is not met, and there are at least two additional files in the current directory that are not directories, so that the first 4 letters in their name are the same as the first 4 letters in the file name, the program will create a new sub-directory whose name is the same as the first 4 letters of the file name, and the file will be moved into this sub-directory.

For example, If the following files exist in the current directory:
file1 file2 file3 prog1 prog2 prog script1 script2,
when only prog is a directory



then, after running the program Arrange.sh, in the current directory will be the following files: file, script1, script2, prog.
- the files file1, file2, file3 will be in the directory **file**.
- the files prog1 and prog2 will be in the directory **prog** in addition to the other files/directories located in prog.



Instructions:
1. Write a function **four_letters** that takes as a parameter word and returns the first 4 letters of this word. For example, **four_letters Natali** will return Nata.
2. Write a function **num_of_files** that takes a word and returns the number of files which start with this word. For example, **num_of_files Nata** will return the numbers of files starting with the word Nata.
3. Use the functions four_letter and num_of_files to finish the exercise.

Submission:

Create one compressed file named Linux_Lab7_id1_id2.tar,
where id1 and id2 must be changed to the id numbers of the two partners.

This file will include the files **MathFunc.sh, Composers.sh, Arrange.sh.**

Submit only the file Linux_Lab7_id1_id2.tar in Moodle and only by one of the partners.

Appeals:

Can be submitted by e-mail to Elad mail@eladhuttner.net within a week from the date of publication of the grades.