Lab11. Shell Scripts (III)- Programming (Looping Constructs)

INSTRUCTOR :MURAD NJOUM

Objectives

After completing this lab, the student should be able to:
- Include programming looping constructs in shell scripts.
- Understand and use the **while, until, and for loops** constructs.
- Learn how to make for loops more efficient by using command outputs as lists.

There are different loop constructs that may be used in shell scripts which include:
while loops
until loops
for loops

Each has its own useful features that make it useful in certain

While Loop

```
while condition
do
statement(s)
done
```

example:

```
vi listarguments
while [ $# -ne 0 ]
do
echo $1
shift
done
:wq
```

Run the above script as follows:
listarguments a hello 7 x
Check the output.

```
Output:
a
hello
7
x
```

Last Delete

Execute the Scripts:

Rewrite the delete script we wrote in the last lab such that it works as follows:

```
delete file1 wrong dir1 file2
File file1 is deleted
wrong: No such file or directory
Directory dir1 is deleted
File file2 is deleted
```

Con t..

Sometimes the loop will stop executing based on the user input, as follows:
***vi* findahmad**

```
echo Enter name
read name
while [ “$name” != “ahmad” ]
do
echo $name: wrong name. Try again.
echo Enter name
read name
done
:wq
```

Question

Now modify the **checkusername** script from the previous lab such that it is called checkusernames instead and works as follows:

```
checkusernames
Enter user name to check or word “enough” to stop
u1112345
Enter user name to check or word “enough” to stop
u11
Enter user name to check or word “enough” to stop
u1123456
Enter user name to check or word “enough” to stop

u1112345 = Salem Hamdi
u11 = No such user name
u1123456 = Sabah Khaled
enough
```

```
echo Enter user name to check or word "enough" to stop
read name
while [ "$name" != "enough" ]
do
y=$(grep^$name  /etc/passwd |cut -d : -f1)
if [ "$name" = "$y"  ]
  then
  x=$(grep^$name /etc/passwd |cut -d : -f5|tr '_' ' ' )
 echo $name = $x
else
  echo No such user name
fi
echo Enter user name to check or word "enough" to stop
read name
done
```

What is the problem with this script??

What about, two user name :mnjoum njoum??

Break and Continue Statements
The programmer can use **break** and **continue** statements inside shell script loops which mean the same as they do in the **C language:**
break - exit the loop immediately.
continue – stop running the current cycle but go back and check the condition.
In addition they can use **break** and **continue** followed by a number to specify how many loop levels they want them to work for. For example:
break 2
Will <u>exit out of two nested loops</u> if they exist.

```
#!/bin/bash
#breaking a loop
num=0
while [ $num -lt 10 ]
do
(( num ++ ))
if [ $num -eq 5 ]
then
echo "break done"
break
fi

echo $num
done
echo Loop is complete
```

```
#!/bin/bash
#continue a loop
num=0
while [ $num -lt 10 ]
do
(( num ++ ))
if [ $num -eq 5 ]
then
echo "continue done"
Continue
fi

echo $num
done
echo Loop is complete
```

until loop
The until loop is similar to the while loop, but stops when the condition becomes true.

```
until false
do
statements
done
```

Modify the above two programs such that they use the until construct instead of the while construct and try them out. Did they work? ______________.

```
until [ "$name" = "enough" ]
```

```
until [ $# -eq 0 ]
```

For loop

In shell scripts, the for loop is very powerful and useful. The general structure of the for loop is as follows:

```
for item in list of items
   do
       statement(s)
   done
```

What makes a for loop powerful is the different ways a list of items may be specified. Let us start with a simple example:

vi names

```
for name in ahmad hamdan subha khaled
   do
     echo $name
   done
```

:wq

Run the script names. It should display the names given in the list

```
#! /bin/bash
for name in $*
do
echo $name
done
```

Rewrite the delete script we wrote at the beginning of this lab such that it uses a for loop instead of a while loop.
Did it work? ________________.

Using a for loop, write a script called ***comp311*** that lists the full names of all the users that are registered in the comp311 course.
Answer:

Now rewrite the script ***comp311*** such that it will display only the names of the users that are currently logged in to the system. (**hint: use the output of the who command**)
Answer:

The for loop can also be applied to a directory of files as follows:

```
vi myfiles
for file in *          or $(ls)
 do
echo $file
done
```

Write a script called ***filetypes*** that uses a for loop to type the name and type (file, dir, or unknown) for each file in a given directory as follows:
Assume that I use the script in the following way:
filetypes /etc
then the script should display the names of all the files under directory /etc and the type of each of those files:

```
#! /bin/bash
for file in $1/*
do
if [ -f $file ]
  then
  echo $file : is File type
elif [ -d $file ]
  then
  echo $file :is Directory type
 else
 echo $file : is Unknown type
fi
done
```

The **which** command displays the directory in the PATH that contains the command. Try
it as follows:
which ls
What is the result? /bin/ls **.**

Write a script called ***mywhich*** that simulates the which command. You are not allowed to use the which command in your script.

(***hint: use the for loop and the sed command***)