//Mohammad hlhwly  
//1191413

//Lab10

Linux Commands General Format

1)Create a file for writing with comments inside.

2)Start writing comments

**Lab10. Shell Scripts (II)- Programming**

**(Selection Constructs)**

**Objectives**

After completing this lab, the student should be able to:

* Include programming selection constructs in shell scripts.
* Use the if/else statement to manipulate integer and string values as well as file properties.
* Apply the case statement programming construct for efficient selections as well as creating menus.

**Script Selection Constructs**

In the previous lab, you have noticed that in our scripts we made several assumptions that files and user names already existed and that we have permissions to remove, copy, or view files and that the correct number of command line arguments where given to our scripts. This is not always the case. Our scripts should be able to check for values and properties before executing what is required. To do this, we need to use selection statements ( the If and Case statements).

Unix commands return a value ( success = zero and failure or error = non-zero) to the shell. This value is stored in the variable (?) as follows:

Run the command:

***ls –al***

Now run the command:

***echo $?***

***What result did you get? \_\_0\_\_\_\_ Why? \_\_\_\_\_\_*** success = zero and failure or error = non-zero ***\_\_\_\_\_\_\_\_\_\_\_\_\_.***

Now run the command:

***cp***

followed by the command:

***echo $?***

***What result did you get? \_\_1\_\_\_\_\_\_ Why? \_\_*** ***There are no files \_\_\_\_\_\_\_\_\_.***

The value returned by Linux commands may be checked in scripts using the if/else structure.

Write the following script:

***vi checkcommand***

***if $1 > out 2> err***

***then***

***echo Command $1 succeeded***

***else***

53

Computer Science Department ( Linux OS Laboratory Manual COMP311 )

***echo Command $1 failed***

***fi***

***:wq***

Now run the script as follows:

***checkcommand date***

***What result did you get? Command [date] succeeded***

***\_ Why? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.***

Now run the command:

***checkcommand mv***

***What result did you get? Command [mv] succeeded***

***\_ Why? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.***

This is one way to use the if/else structure. Still, many scripts do not check commands, but rather check for variable values, file properties, and number of arguments. To do that we need to use one of two syntaxes:

***if test*** ***condition ( e.g.***

or

***if [ condition ] ( e.g. if [***

***if test $# -eq 2 )***

***$# -eq 2 ] )***

The general syntax for the if/else statement is as follows:

***if condition***

***then***

***statements***

***elif condition***

***then***

***statements***

***else***

***statements***

***fi***

To compare integer values, we use the following relational operators:

***-lt (less than), -gt (greater than) -eq (equal)***

***-le (less than or equal) -ge (greater than or equal), -ne (not equal).***

Let us rewrite the delete script we wrote in the previous lab to check for the correct number of arguments as follows:

***vi delete***

***if [ $# -eq 1 ]***

***then***

***rm $1***

***echo $1 is deleted***

***exit 0***

***else***

***# This line returns 0 from the script (success)***

54

Computer Science Department ( Linux OS Laboratory Manual COMP311 )

***echo Usage: delete filename***

***exit 1***

***fi***

***:wq***

Now try the above script as follows:

***delete myfile*** (assuming myfile exists and is a regular file)

Then run the command:

***echo $?***

***Did it work?\_Yes ---\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.***

***What is the value of variable (?) ?\_\_\_ 0 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Now try it as follows:

***delete***

Then run the command:

***echo $?***

***What happened? \_\_\_*** ***nothing \_print : Usage: delete filename \_\_\_\_\_\_ Why?\_*** ***there's no file \_\_.***

***What is the value of variable (?) ?\_1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

To check file values we use the following operators:

***-f filename*** ( to check if file exists and is of type file)

***-d filename*** ( to check if directory exists and is of type directory)

***-x,-r,-w*** (to check if a user has execute, read, or write permissions on a file)

Let us rewrite our delete script to include those:

***vi delete***

if [ $# -eq 1 ]

then

if [ -f $1 ]

then

rm $1

echo File [$1] is deleted

exit 0

elif [ -d $1 ]

then

rm -r $1

echo Directory [$1] is deleted

exit 0

else

echo no file or directory name [$1]

exit 1

fi

else

echo [$1] file or directory

exit 2

fi

55

Now create a file and a directory using the following commands:

***touch myfile; mkdir mydir***

No try the updated delete script in the following ways:

***delete***

***What happened? \_*** [] file or directory***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.***

***delete*** ***myfile*** ( myfile exists and is a file )

***What happened?\_\_\_\_*** File [myfilee] is deleted***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.***

***delete mydir*** ( mydir exists and is a directory)

***What happened?\_\_\_*** directory name [mydir]is delete***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.***

delete wrong ( wrong does not exist )

***What happened?\_\_\_\_\_\_\_*** no file or directory name [wrong]***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.***

***Now rewrite the copy script to act as follows:***

***copy***

***Usage: copy src*** ***dest***

***copy myfile newfile***

***File myfile is copied to file newfile***

***copy mydir*** ***newdir***

***Directory mydir is copied to newdir***

***copy wrong good***

***wrong: No such file or directory***

.

if [ $# -ne 2 ]

then

echo Usage : cp src dst

exit 1

elif [ -f $1 ]

then

cp $1 $2

echo File [$1] copied to [$2]

exit 0

elif [ -d $1 ]

then

cp -r $1 $2

echo Directory [$1] copied to [$2]

exit 0

else

echo No such file or directory name [$1]

exit 2

fi

fi

Try the new copy script and make sure it works as above?

***Did it work correctly? \_\_Yes\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.***

Sometimes our scripts need to check string values. To do that we need to use the following operators:

***= (equal), != (not equal)*** ***-n (none null string) -z (zero string (null))***

Let us try some of those. let us write a script to check the value of the name entered by the user:

***vi checkname***

56

Computer Science Department ( Linux OS Laboratory Manual COMP311 )

***if [ $# -ne 1 ]***

***then***

***echo Usage: checkname*** ***name***

***exit 1***

***else***

***if [ “$1” = “ahmad” ]***

***then***

***echo $1: Hello***

***exit 0***

***else***

***echo $1: Goodbye***

***exit 0***

***fi***

***fi***

***:wq***

try it as follows:

***checkname*** ***ahmad***

What happened?\_\_\_ahmad Hello\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

***checkname*** ***suha***

What happened?\_\_\_\_\_ ***suha Goodbye*** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

***checkname***

What happened? ***Usage: checkname*** ***name***

\_\_\_\_\_\_\_\_\_\_\_.

***Write a script called checkusername which works as follows:***

***checkusername***

***No names were entered***

***checkusername u1112233***

***u1112233 = Ahmad Hamdan***

***checkusername u11***

***u11 = No such user name***

***checkusername bash***

***bash = No such user name***

Script:

if [ $# -ne 1 ]

then

echo Usage : checkusername

exit 1

else

output=$(cat /etc/passwd | grep "^$1:")

if [ -z "$output" ]

then

echo [$1] = no such username

exit 2

else

fullname=$(cut -d: -f5 $output)

echo [$1] = $fullname

exit 0

fi

fi

57

Computer Science Department ( Linux OS Laboratory Manual COMP311 )

***Now try it with similar cases to those written above.***

***What happened?\_*** ***It worked \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.***

**Case Statement**

We can also use a case statement ( similar to switch in c) to check for values. The syntax is as follows:

***case*** ***value*** ***in***

***pattern1) statements***

* ***# ;; is the break statement***

***pattern2) statements***

***\*)*** ***statements***

***esac***

# \* stands for anything which is the default case

The patterns may be strings or parts of strings. Those can include the \* wild card, the (|)

OR operator, as well as ranges (e.g [0-9] or [a-f]) as follows:

s\* | S\* | good)

means any pattern that starts with s or S or the word good.

[A-Z]\*[0-5])

means any pattern with any size that starts with a capital letter and ends with a number between 0 and 5

[a-z][0-9][0-9][0-9] | [0-9][A-Z][A-Z][A-Z][a-f])

means the accepted pattern must consist of exactly four characters the first is a small letter and the next three are numbers or the pattern must be exactly five characters with the first being a number followed by three capital letters and then one small letter between a and f.

case $1 in

s\* | S\* | good)

echo pattern1

;;

[A-Z]\*[0-5])

echo pattern2

;;

[a-z][0-9][0-9][0-9][0-9][A-Z][A-Z][A-Z][a-f])

echo pattern3

;

; \*)

echo no pattern found

;;

esac

***Write a script that uses case statement with patterns similar to the above.***

***Did they work?\_\_\_Yes\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.***

Case statements are usually used for handling menus and menu options. Let us try a simple example that uses a menu to call different scripts (modular programming):

Create three different scripts called ***script1***, ***script2***, and ***script3*** respectively. In each script put one line to display which script you’re in (e.g in script1 put the line “echo this is script 1”).

Now create a script called ***mainscript*** that displays the following menu:

***Please select your choice (1-4):***

***1 - Run script1***

***2- Run script2***

***3- Run script3***

***4- Exit main script***

58

Computer Science Department ( Linux OS Laboratory Manual COMP311 )

***Using a case statement, have your script run the suitable script (1,2, or 3) or exit based on the user’s selection.***

***vi mainscript***

echo please select your choise :

echo 1- Run script 1

echo 2- Run script 2

echo 3- Run script 3

echo 4- Exit

read choise

case $choise in

1)script1

;;

2)script2

;;

3)script3

;;

4) exit 1

;;

\*) echo invalid choise

;;

esac

Now try ***mainscrip***t. ***Did it work?\_\_\_Yes\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.***

3) Exit to save all lab workers