Shell Scripting 1

The use of quotes in grep command:

Description: Searches for patterns in files.

Usage: grep [options] pattern [file(s)]

Examples:

- grep Fadia addresses.txt
- grep 'Salem Kaseer' addresses.txt
- grep Salem addresses.txt
- grep Salem Kaseer addresses.txt // this will give you an error

what you need to know about quotes :

1. if you want to search for a string contains two or more words you have to use the single quotes.
2. The single quotes ' ' ignores every thing inside it.
3. So there is another option which is the Double quotes " "
4. Where they are not restricted like the single one , Double quotes doesn't ignore the following signs :
   - Dollar signs ($)
   - Back quotes (')
   - Back slashes (\)
5. The Back slash sign is same as single quotes.

PATH environment variable:

Description: Specifies directories in which the shell looks for executable files.

Usage: PATH=$PATH:/path/to/directory

Example:

- PATH=$PATH:/home/user1/shell

mv command:

Description: Moves or renames files and directories.

Usage: mv [options] source destination

Example:

- mv $filename ${filename}X

Notes:

You can apply arithmetic expressions on Integers like this : $((expression)) , and it can be used with echo command.

But of course there is a better option which is the expr command.

expr command:

Description: Evaluates expressions.

Usage: expr [options] expression

Examples:

- expr 1 + 2
- expr 5 * 5 // you can use the multiply operation only in this expression : ‘ * ’.
- expr 10 + 20 / 2
- expr 10 - 9

Command Substitution:

Usage: to pretty up the text 😊

Example:

- echo “the date is : `date`” // this will give you the following output : the date is : current date . Note that we’ve used the back quotes.

echo command (inside a script):

Example:

- echo "$1 $2" >> phonebook

grep command (inside a script to remove):

Usage: grep -v "$1" phonebook > /tmp/phonebook

Example:

- grep -v "$1" phonebook > /tmp/phonebook

mv command (inside a script to remove):

Usage: mv /tmp/phonebook phonebook

Example:

- mv /tmp/phonebook phonebook

Passing Arguments to shell scripts:

Importance: **Shell programs become more useful when you learn how to process arguments passed to them.**

How it works :

After defining a certain amount of variables in the shell file you can pass a value for them when you execute the file as follows :

- ./file.sh 1 2 3

$# command (inside a script to display the number of arguments):

Example:

- echo $# arguments passed

$* command (inside a script to read arguments from user <any number of args>) :

Example:

- echo they are :$*:

Passing more than 9 arguments to shell scripts:

If you provide more than nine arguments to a program, accessing the tenth and higher arguments with $10, $11, etc., won't work as expected. Instead of representing the tenth argument, for instance, $10 would substitute the value of $1 followed by 0.

To access these arguments beyond the ninth, enclose them in curly braces {}. For instance, to access the 11th argument, the script should use ${11}.

shift command:

Description: Shifts command line arguments.

The shift command allows you to effectively left shift your positional parameters. If you execute the command: shift whatever was previously stored inside $2 will be assigned to $1, whatever was previously stored in $3 will be assigned to $2, and so on. The old value of $1 will be lost. In addition, $# (the number of arguments variable) is also automatically decremented by one.

Example:

- shift
- shift 3

echo command (to display exit status):

Example:

- **echo $?**

rm command (inside a script to remove a file):

Usage: \rm fileNoExist

Example:

- **\rm fileNoExist**

Notes:

You can use all of the commands in the last 3 experiments in a shell file instead of using them in the terminal.

To create the shell file do this:

- **Touch filename.sh**
- **chmod +x filename.sh // to give the permission to execute**
- **to execute the file use ./filename.sh**

- you can use # to write comments in the shell files like this : # this is a good comment.

- to create a variable do like this var=56 // very important note : Do not put spaces before or after the equal sign

- there is no data types in shell, so you can assign what ever you type of data to the variable.

- If you didn't give the variable any value you will not face any problem, put when you print the variable you will get an empty line , you can also use these methods to assign a null value to the variable : - dataflag1="" or - dataflag2=''
- to use the variable in the file do this : echo $var

اللَّهمَّ أنتَ ربِّي، لا إلَهَ إلَّا أنتَ، خَلقتَني وأَنا عبدُكَ، وأَنا على عَهْدِكَ ووعدِكَ ما استطعتُ، أعوذُ بِكَ من شرِّ ما صنعتُ، وأبوءُ لَكَ بنعمتِكَ عليَّ، وأعترفُ بِذنوبي، فاغفِر لي ذنوبي إنَّهُ لا يَغفرُ الذُّنوبَ إلَّا أنتَ.