Ashesi University College: Networks Lab 5 b

Part IIb. Creating shell scripts

Shell scripts are collections of commands that may be executed in a batch. A **shell script** is a sort of program to be run by the shell or command line interpreter. It may be just a collection of commands, but can include control logic. Shell scripts can be made to manipulate file system, start other program, or do anything that a user would usually do from commad line. It treats other programs as functions and can read their return value.

A. Learn to create a basic hello work shell script.
Using your preferred editor, create a file **hello.sh** in your home directory.
Use the following content:
First line of every script is #!/bin/bash. Note that there are no semicolons at the end of statements, but can put multiple commands on same line separated by semi colons.
Comments start with #, but the first line is not a comment.

```
#!/bin/bash
echo "Hello work script"
who
date
```

save the file and make it executeable with: chmod 755 hello.sh
or chmod +x hello.sh
Run it from command prompt as: ./hello.sh
Note: echo without going to next line is echo –n "message"

Tutorial:

Variables:

```
X=44
Arithmetic: let y=$x+1          #y is 2
String: z= $x+1                 #z is 1+1
Echo "Value of x is $x"
```

Some preset values: $USER, $HOME, $PATH
Command line argurements are $0, $1, $2, … where $0 is name of script itself.
$# number of arguments given
$* list of all arguments given
You can thus write: if [$# == 3]

Control structures

if

```
if [ condition ]; then
   command(s)
  elif [ condition ]; then
   command(s)
  else
```

command(s)
fi

note the space around the condition!

Examples: [i]

```
#string
var=adam
if [ $1 = $var ] ; then echo "string $1 equals $var" ; fi
if [ $1 == $var ] ; then echo "string $1 equals $var" ; fi
if [ $1 != $var ] ; then echo "string $1 does not equals $var" ; fi
if [ -z “$1” ] ; then echo "string $1 is empty!"; fi
if [ -n $1 ] ; then echo "string $1 is not empty!"; fi

#numeric
a=1
if [ $a -eq $1 ] ; then echo "number $1 equals $a" ; fi
if [ $a -ne $1 ] ; then echo "number $1 does not equal $a" ; fi
if [ $a -gt $1 ] ; then echo "$a is greater than $1" ; fi
if [ $a -lt $1 ] ; then echo "$a is less than $1" ; fi

#file/dir properties
if [ -d $1 ] ; then echo "$1 exists and is a directory!" ; fi
if [ -e $1 ] ; then echo "$1 exists!" ; fi
if [ -f $1 ] ; then echo "$1 exists and is not a directory!" ; fi
if [ -r $1 ] ; then echo "$1 exists and is readable!" ; fi
if [ -s $1 ] ; then echo "$1 exists and has size greater than zero!" ; fi
if [ -w $1 ] ; then echo "$1 exists and is writable!" ; fi
if [ -x $1 ] ; then echo "$1 exists and is executable!" ; fi
```

While:

while *test*
do
command(s)
done

eg

```
i=1
while [ $i -le 10 ]
do
   echo $i
   let i++ # you can use incrementers in lets
         # and, you don't need to use the $ here
done
```

For

for *varname* in *value1 value2 ...*
do
command(s)

```
  done
eg
i=1
for arg in $*
do
   echo "arg $i: $arg"
   let i++
done
```

Case

```
case expression in
   pattern1 )
      statements ;;
   pattern2 )
      statements ;;
   ...
   *) the_Default;;
esac
```

Note use of double semicolon, use of) for each case, *) for default, ends with esac
Eg (this example uses multiple case options)

```
#!/bin/bash
NOW=$(date +"%a")
case $NOW in
        Mon)
                echo "Full backup";;
        Tue|Wed|Thu|Fri)
                echo "Partial backup";;
        Sat|Sun)
                echo "No backup";;
        *) ;;
esac
```

Doing I/O

Use read to accept input eg thefollowing will ask for user to input a value

```
Echo “enter a filename”
read aname
echo “name entered is $aname”
```

Example scripts

<see separate file>

Lab work:

Write a script that accepts two parameters as follows:
First parameter is your user name, and second is your file you would like to create.
If the script is run without two parameters, it should display a message that says:
Usage: scriptname <username> <filename>
The script should do the following:

- display a greeting to the current user, using their name
- print the current date
- List all files in the /home/tester directory and save then in a file in your own home directory using the filename the user supplied.
- Create a script (give it a name you prefer) whose job is to be an alias of the command "ls –l /home | grep nama" This means if your new script is called see.sh, anytime user types see, they will get a listing of some home directories.
- Use the tar command with a gzip option to create a new archive file. It must contain the two files (the previous file saved as well as your script)

To be evaluated next class.

- Copy the archive file created to /home/namanquah/public_html/nwlab05, but remane it at the destination to your username with the digit two appended. (this means if I created a file called data.tar.gz, it will be saved at the target location as namanquah2.tar.gz
- Your program must execute /namanquah/mysteryscript. Pass it your username as a parameter.

Manual operation: (you script should not do this)
Name your script by your username appended with 1, and with a .sh extension eg namanquah1.sh. Copy the script to /home/namanquah/public_html/nwla05. Make your script have the following permissions: -rwx—x—x

You will have two files at the end of the lab: namanquah1.sh and namanquah2.tar.gz in /namanquah/public_html/nwlab05/. These two will be graded.

Dr N Amanquah

---

[i] http://www.usna.edu/Users/cs/aviv/classes/ic221/s14/lab/02/lab.html