Shell Scripting

Table of Contents

Shell Scripting

1) #!/bin/bash → (shebang)

- always mention shebang in first line of script so as to use bash shell or else default shell will be used
- if we didn't mention shebang in first line of script then all the commands will be executed in default shell
    - bash shell → default shell in remote server
    - default shell → not mentioning shebang

    - echo $SHEL → to check default shell running in the system
    - chsh bash → to change to other shell

2) script1.sh

(.sh is extension for shell script)

#!/bin/bash (vi terminal)
echo "This is Devops Class" → echo used to print
echo "Started on 08th May 2024"

> ***Methods to use for execution*** (in Linux terminal)
> ./script1.sh → mostly used
> bash script1.sh
> sh script1.sh

3) script2.sh

```
#!/bin/bash
echo "This is $1 Class"          #($ = variable, input arguments)
echo "Started on $2"

./script2.sh DevOps 31May24      #after 1 space 1st argument is considered followed by 2nd argument
```

> **Input arguments:**
> i/p arguments from 1 to 9
> we mention $1, $2, $3, $4.... $9
> from 10th onwards
> ${10}, ${11}, ${12} ...
>
> $1 --> first input argument
> $2 --> second input argument
> $3 --> third input argument
>
> ''''
> ''''
> $9 --> ninth i/p argument

4) script3.sh: (declaring var and passing it)

```
#!/bin/bash
var1="devops" → variable name = variable value
age=30
num="2 4 6 8 10"
echo "This is $var1 and the age is $age"
echo "$num are even"
```

if statement:

Syntax: if [condition]
then
action
else
action
fi

5) script4.sh: (if else)

```
#!/bin/bash
if [ $1 -eq 5 ]
then
echo "$1 is equal to five"
else
echo "$1 is not equal to five"
fi
```

logical operator/parameters

-eq → equals
-lt → less than
-le → less than or equal
-gt → greater than
-ge → greater than or equal
-ne → not equal]

set -x → to debug the shell script, shows detailed output of commands

AND
2^2 = 4
2^3 = 8

A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

AND

A	B	o/p
0	0	0
0	1	0
1	0	0
1	1	1

OR

A	B	o/p
0	0	0
0	1	1
1	0	1
1	1	1

NOT

A	o/p
0	1
1	0

6) *script5.sh: (Biggest of 3 no)*

$1=9
$2=5
$3=15
} arguments

```
#!/bin/bash
if [ $1 -gt $2 -a $1 -gt $3 ]
then
echo "$1 is Big"
elif [ $2 -gt $3 -a $2 -gt $1 ]
then
echo "$2 is Big"
else
echo "$3 is Big"
fi
```

7) script6.sh: (restricting arguments)

```
#!/bin/bash
if [ $# -ne 2 ]                    # restricting to 2 arguments (-ne = not equal to)
then
echo "Enter only two input arguments"
elif [ $1 -gt $2 ]
then
echo "$1 is Big"
else
echo "$2 is Big"
fi
```

Special characters of global variable

$? → used to check status of last executed command, output showing 0 means success and if output is non-zero means failure

0 → success, non-zero → failure

$$ → PID (Process ID) of current running process

$! → PID of last command execution went into background

$* → all arguments

$@ → all the arguments pass to shell script stored in an array format
[ajay vijay test names] [12 13 14 15 16]

$# → total number of arguments pass to a shell script

8) nohup (no hangup):

- used to make script run in background so as we can work on other scripts as terminal will be available using nohup or else terminal won't be available until script is executed in front ground. When we use nohup then script will be stored in nohup file and Process ID(PID) will be shown.
- nohup will always append output instead of overwriting previous outputs.

(***Syntax:*** nohup ./script.sh input &
cat nohup.out → check output in file)

9) bg (background:

nohup is used before executing any scripts, we can't use it when script is running instead, we can use bg(background) which can be used at any time so as to make script run at background.

(***Syntax:***
Step1: Ctrl+Z → stop but doesn't exit
Step2: bg → script will be sent to background
ps -ef → displays scripts running in background
ps -ef | grep 1339(PID) → to check any scripts using PID)

[**Note:** When we use bg, script won't get stored in any file like nohup instead will be running in background and once it's executed then it will be automatically displayed in terminal]

10) fg (foreground):

by using fg if any scripts are sent to background for execution, then it will come back to foreground(terminal) and start executing in terminal.

11) script7.sh: (addition, sub, mul, div of two number)

```
#!/bin/bash
sum=`expr $1 + $2`
mul=`expr $1 \* $2`
if [ $1 -gt $2 ]
then
sub=`expr $1 - $2`
div=`expr $1 / $2`
else
sub=`expr $2 - $1`
div=`expr $2 / $1`
fi
echo "sum of $1 and $2 is $sum"
echo "mult of $1 and $2 is $mul"
echo "sub of $1 and $2 is $sub"
echo "div of $1 and $2 is is $div"
```

if [condition]
then
statement
else
statement
fi

==========

 # use when having anything in list/array

for i in list **do** **commands** **done**	**[2 4 6 8] → list** **sum=20** **for i in list** **add= `expr $sum + $i`** **done**

==========

 # can use for larger no, smaller no, greater than, etc...
while [condition]
do
commands
done

12) script8.sh: (sum of given set of numbers (5 4 3 2 1))

```
#!/bin/bash
echo "Enter a number for sum: "
read num
sum=0
while [ $num -gt 0 ]
do
sum=`expr $sum + $num`
num=`expr $num -  1`
done
echo "Sum of given number is $sum"
```

13) script9.sh: (Decimal value)

```
#!/bin/bash
sum=`expr $1 + $2`
mul=`expr $1 \* $2`
sub=`expr $1 - $2`
div=$(echo "$1 / $2"|bc -l)
echo "sum of $1 and $2 is $sum"
echo "mult of $1 and $2 is $mul"
echo "sub of $1 and $2 is $sub"
echo "div of $1 and $2 is $div"
```

14) script10.sh: (factorial of given no)

```
#!/bin/bash
#set -x
echo "enter number"
read num
fact=1
counter=1
while [ $counter -le $num ]
do
fact=`expr $counter \* $fact`
counter=`expr $counter + 1`
done
echo "The factorial of $num is $fact"
```

15) script11.sh: (write a script to count the number of words and characters in each line from a given file: (countwordchar))

```
#!/bin/bash
#set -x
num=0
while read line                                    #reads lines line by line
do
words=`echo "$line"|wc -w`                         # (w = count words)
num=`expr $num + 1`
```

```
char=`echo "$line"|wc -m` `                    # (m = count characters)
echo "line number $num: total words $words: total char $char"
done<$1                                        #read line takes input from here
```

16) script11.sh: write a script to check given name is a file/directory

i) if it's a file displays the file content

ii) if it's a directory list only the files present inside of it

```
#!/bin/bash
NAME="$1"
if [ -f "$NAME" ]
then
   # If it's a file, display its content
   echo "Displaying content of the file '$NAME':"
   cat "$NAME"
else [ -d "$NAME" ]
   # If it's a directory, list only the files inside it
   echo "Listing files in the directory '$NAME':"
   ls "$NAME"
fi
```

17) script12.sh: list all the files, if the files containes the pattern word "Linux", if none of the files containing linux pattern, display the message "Pattern not found in any of the files"

```
#!/bin/bash
pf=$(basename "$0")
files=$(grep -ilR --exclude="$pf" "linux")
if [ "$files" ]
then
echo "files which contains pattern linux
are $files"
else
echo "files not found"
fi
```

or

```
#!/bin/bash
output=`grep -ilr "$pattern" *`
if [ $? -ne 0 ]
then
echo "The given pattern not found in any of
the files"
else
echo "The below files will contain $pattern"
echo $output
fi
```

18) script13.sh: write a script to check given name is a file/directory

i) if it's a file displays the file content

ii) if it's a directory list only the files present inside of it

```
#!/bin/bash
echo "Enter the name to check: "
read name
if [ -f $name ]
then
echo "Given $name is a file, Content of file is"
cat $name
elif [ -d $name ]
then
echo "$name is a directory, list of files is:"
find $name -maxdepth 1 -type f
else
echo "$name is not found"
fi
```

19) script14.sh: check the give name is a file or directory or link

```
#!/bin/bash
echo "Enter the name to check: "
read name
if [ -h $name ]
then
echo "$name is a softlink"
elif [ -f $name ]
then
echo "Given $name is a file"
elif [ -d $name ]
then
echo "$name is a directory"
else
echo "$name is not found"
fi
```

20) script15.sh: Storage alert script:

Cron job/cron tab:

*** → Min (00-59)**
*** → Hours (00-23)**
*** → date (01-28/29/30/31)**
*** → Month (1-12)**
*** → Day of the week (00-06)**

* * * * * /home/ubuntu/script.sh
00 * * * * /home/ubuntu/script.sh
5th sep 9:30AM Tuesday
30 09 05 09 02 /home/ubuntu/script.sh

```
#!/bin/bash
space=`df -h .|tail -1|awk -F " " '{print $(NF-1)}'|sed 's/%//g'`        #(df -h=displays size consumed by drive, tail -1=displays last line, awk..(NF-1)=cuts and displays last second column separated by space, sed…..=replace pattern name)
if [ $space -ge 90 ]
then
echo "Disk is full, please take the action" | mail -s "Subject" -c "devops.com" test@gmail.com        #(send mail to cc mail ids and mail id)

fi
00 * * * * /home/ubuntu/memory.sh        # crontab
```

21) script16.sh: Service status alert script

```
#!/bin/bash                                  #(shebang)
services="tomcat java apache2"               # (services as example)
for i in $services                           # (since services are in list, we use for loop)
do
sudo service $i status                       #(checks service status)
if [ $? -ne 0 ]                              #($?=checks status of last executed command notequalto 0 as 0=success, non-zero=failure)

then
echo "The $i service is stopped, please take the action"|mail -s "Service Stopped" -c "group@mail.com" team@mail.com        #(mail s=subject, c=cc)
fi
done
```

22) script18.sh: Cleanup script:

```
#!/bin/bash                                  #(shebang)
total=`ls|wc -l`                             #(ls=list, wc -l=word count lines)
delete_count=`expr $total - 20`              # (delete_count=variable, keep 20 files)
if [ $delete_count -gt 0 ]
then
ls -rt|head -$delete_count|xargs rm -r       #(ls -rt=list as per time in reverse order, delete all files except 20 latest files)

fi
```