Bash Scripting Presentation – Ubuntu-MD Nov 23, 2019

Bash shell scripting allows you to run any terminal command in a program without having to compile it or install any other program. Bash comes pre-installed as part of Unix, Linux and Apple OS operating systems.
Variables have no scope, they are Global, there is no standard libraries and you don't have a modular system. You can perform network requests with wget and process text using awk and more.
Scripts are stored in files. The she-bang command #!/bin/bash should be the first line. Bash script files should be executable, chmod +x {filename}. I also like to add the extension, sh to my script files.
You can use an online terminal to test scripts like https://rextester.com/l/bash_online_compiler but most use the Terminal on their device.
Comments in a script start with # symbol.

```
#!/bin/bash
# this is a comment
```

Variables are assigned using the = symbol
name=value
room_no=250
name="Ron"
To print use the echo command and add $ to the variable name
echo $room_no (return 250)
echo $name (return Ron)

Operators
+, -, *, / #Add variables sum_no=$((num1+num2))
<, <=, >, >=
-lt (lower than)
-gt (greater than)
-le (lower or equal than)
-ge (greater or equal than)
-eq (equal to)
-ne (not equal to)
&& logical And
|| logical Or

Example

```
#!/bin/bash
age=23
minimum=18
if test $age -lt $minimum
then
  echo "Not old enough"
fi
```

```
If/Else Statements
      if condition
      then
        command
```

```
      else
        anothercommand
      fi
```

If/Else/Condition Statement

```
      if condition
      then
        command
      elif
        anothercommand
      else
        yetanothercommand
      fi
```

Example (if/else statement)

```
#!/bin/bash
DOGNAME=Roger
if [ "$DOGNAME" == "" ]; then  ## brackets are required for Boolean statement
  echo "Not a valid dog name!"
else
  echo "Your dog name is $DOGNAME"
fi
```

Loops

While Loops
While condition resolves to a true value, run command

```
      while condition
      do
        command
      done
```

Reading user input with read

In many ocations you may want to prompt the user for some input, and there are several ways to achive this.
This is one of those ways:

```
#!/bin/bash
echo Please, enter your name
read NAME
echo "Hi $NAME!"
```

As a variant, you can get multiple values with read, this example may clarify this.

```
#!/bin/bash
echo Please, enter your firstname and lastname
read FN LN
echo "Hi! $LN, $FN !"
```

Getting the return value of a program

In bash, the return value of a program is stored in a special variable called $?. This illustrates how to capture the return value of a program, I assume that the directory dada does not exist.

```
#!/bin/bash
cd /dada &> /dev/null
echo rv: $?
cd $(pwd) &> /dev/null
echo rv: $?
```

Get Date Parse

```
#!/bin/bash
Year=`date +%Y`
Month=`date +%m`
Day=`date +%d`
Hour=`date +%H`
Minute=`date +%M`
Second=`date +%S`
echo `date`
echo "Current Date is: $Day-$Month-$Year"
echo "Current Time is: $Hour:$Minute:$Second"
```

Sleep Example

```
#!/bin/bash

echo “Wait for 5 seconds”
sleep 5
echo “Completed”
```

Test if File Exists

```
#!/bin/bash
filename=$1
if [ -f "$filename" ]; then
echo "File exists"
else
echo "File does not exist"
fi
```

Make Directory If It Dose Not Exist

```
#!/bin/bash
echo "Enter directory name"
read ndir
if [ -d "$ndir" ]
```

```
then
echo "Directory exist"
else
`mkdir $ndir`
echo "Directory created"
fi
```

Add Numbers with Double ((

```
#!/bin/bash
echo "Enter first number"
read x
echo "Enter second number"
read y
(( sum=x+y ))
echo "The result of addition=$sum"
```

Swift-Nolapro Backup

```
#!/bin/bash
# Backup backoffice and nolapro website files
NOW=$(date +"%d-%m-%Y")
file1="/home/ron/backoffice-$NOW-.tgz"
file2="/home/ron/nolapro-$NOW-.tgz"
echo "Backing up backoffice..."
tar -czf $file1 /var/www/backoffice.swiftstaffing.com

echo "Backing up nolapro site..."
tar -czf $file2 /var/www/nolapro/
```

Swift-Nolapro Restore Program

```
#!/bin/bash
#Transfer files from *.02 and 108 with date name
past=$(date --date="yesterday" +"%d-%m-%Y")
swiftpgsl=swift-"$past".sql
nolapro2=nolapro-"$past".sql
swifttime=swifttime-bkup.sql
echo $swiftpgsl
echo $nolapro2
```

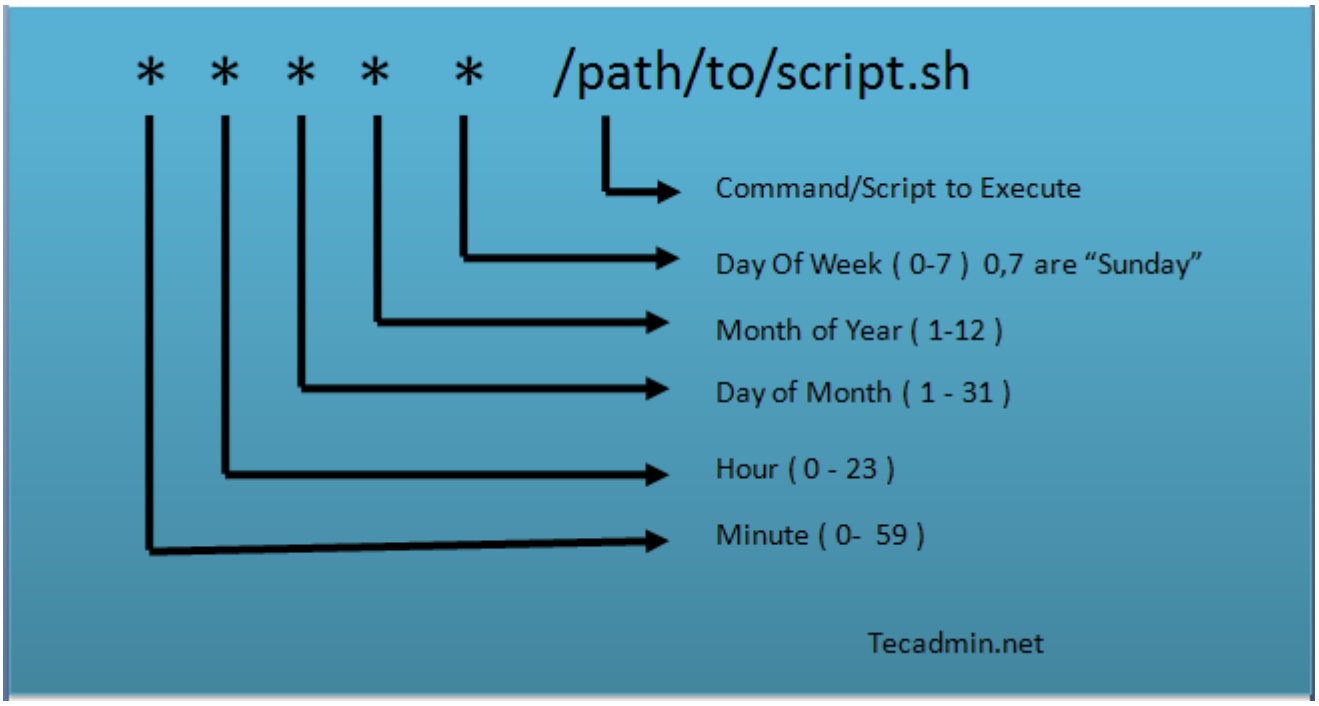
```
#scp 192.168.0.2:/home/ron/"$swiftpgsl" /home/ron
#scp 192.168.0.2:/home/ron/"$nolapro2" /home/ron

#Call update program
sh swift-postgre-restore.sh "$swiftpgsl" "$nolapro2" "$swifttime"
```

Swift-Nolapro Database Update

```
#!/bin/sh
pgfile="$1"
myfile="$2"
timefile="$3"

#scp files
scp 192.168.0.2:/home/ron/"$pgfile" /home/ron
scp 192.168.0.2:/home/ron/"$myfile" /home/ron
scp -P7223 192.168.0.108:/home/ron/"$timefile" /home/ron

#Drop database swift
dropdb swift

#Recreate database swift
createdb -O ron swift

#Restore from backup
psql -U ron swift < "$pgfile"

#Restore nolapro database
echo Restoring nolapro database...
mysql -u root -pxxxxx nolapro < "$myfile"

#Restore SwiftTime database
echo Restoring SwiftTime database...
mysql -u root -pyyyy SwiftTime < "$timefile"
```

Cron Job

Cron is the scheduling application that lets you run your bash scripts or other programs at various times. You edit the crontab program or place the bash script in /etc/cron.daily, /etc/cron.hourly, /etc/cron.weekly or /etc/cron.monthly. You can also be very time specific by using the crontab program parameters:




```
sudo crontab -e
```

Examples

```
0 5 * * * home/someone/test_bash.sh #This would run 5 am everyday

30 23 1 * * /path_to_file/any_bash.sh #This would run 11:30 pm first day of every month

0 10 15 1-6 * file.sh # Runs 10 am on 15 of the month in Jan-Jun
```

Examples

```
#!/bin/bash

## if_with_or.sh
echo "Enter any number"
read n

if [[ ( $n -eq 15 || $n  -eq 45 ) ]]
then
echo "You won the game"
else
echo "You lost the game"
fi
```

```
#!/bin/bash
## substring_example.sh
Str="Learn Linux from LinuxHint"
subStr=${Str:6:5}
echo $subStr
```

Returns Linux

```
#!/bin/bash
echo "Enter Your Name"
read name
echo "Welcome $name to Linux World"
```

```
#!/bin/bash

# This bash script is used to backup a user's home directory to /tmp/.
## basic-backup-home.sh

user=$(whoami)
input=/home/${user}/Documents
output=/tmp/${user}_home_$(date +%Y-%m-%d_%H%M%S).tar.gz

tar -czf $output $input 2> /dev/null
echo "Backup of $input completed! Details about the output backup file:"
ls -l $output
```

```
#!/bin/bash

##function_return.sh
function greeting() {

str="Hello, $name"
echo $str

}

echo "Enter your name"
read name

val=$(greeting)
echo "Return value of the function is $val"
```

```
Returns Return value of the function is Hello, name entered.
```

```
#!/bin/bash
## loops_bash.sh

for i in 1 2 3; do
    echo $i
done
```

```
#!/bin/bash

greeting="Welcome"
user=$(whoami)
day=$(date +%A)

echo "$greeting back $user! Today is $day, which is the best day of the entire
week!"
echo "Your Bash shell version is: $BASH_VERSION. Enjoy!"
```