FROM:
GAURAV SHARMA
YOUTUBE

SHELL SCRIPTING

26-JUN-22
10:27PM
SUNDAY

FIRST SHELL SCRIPT:-

Create a file like firstscript.sh
give it execule permission and run that script.

vi firstscript.sh
chmod +x firstscript.sh
./firstscript.sh or # bash firstscript.sh

- To check available shell in your system.

cat /etc/shells.

- To extract shell or python you may use

which python3 or # which

VIDEO 5:
COMMANDS / KEYWORDS / BUILT IN

How we can check that it is a command or keyword or a built in by

type -a uptime Command
type -a echo or pwd built in

Type -a if or ~~cmd~~ key word

If any thing is not a shell boiltin then it will search in $path and then execute

VIDEO 6:-

PRINT IN DIFFERENT COLOR IN SHELL SCRIPT

```
echo -e "\033[0;32m success msg here"
echo -e "\033[0;31m another msg"
echo -e "\033[0;33m one other msg"
```

VIDEO 7:

COMMENTS & ECHO

- Use # for comments
- If use '\' backslash it will remove space and consider a single line. like

```
echo " my \
name \
is \
zohaib "
```

Output:

my name is zohaib

Single Inverted Commas ' ' : STRONG QUOTE

If strong quote is used with echo it will print as it is and bring no change on output.

- \t introduce a space or tab in blw words
- \v vertical tab
- \n new line

VIDEO 8:-

USER DEFINE VARIABLE

That variables which is defined by user

example

name = zohaib
age = 35
echo "my name is ${name} and age ${age}".

VIDEO 9

VALID VARIABLE:

- _VAR
- VAR-2
- VAR2NAME

INVALID VARIABLES:-

- 3VAR
- @VAR
- VAR NUMER

Not a reserve word

VIDEO 10 :-

SYSTEM DEFINE VARIABLE

Type 'env' to show system define variable

example

```
echo ${SHELL}
echo ${HOME}
echo ${$}      # process id
echo ${PPID}   # parent process id.
```

VIDEO 11 :-

USER INPUT (READ)

```
#read name
#read age
```

For prompt use "-p" with read

For passwords or secrets use "-s"

```
# read -p "please enter your password" -s password
```

VIDEO 12 :-

COMMAND LINE ARGUMENTS :-

To use command line argument you need to give arguments along with commands

./commandlineargu.sh zohaib BS

Example:-

```
#! /bin/bash
name=${1}
age=${2}
echo "my name is ${1} and my age is ${2}"
```

METHOD 1: of VIDEO 14:-

ASSIGN A COMMAND OUTPUT TO A SHELL

To save output to a variable by single back-tic `pwd`

example:

```
CURRENT_WORKING_DIR=`pwd`
```

METHOD 2:- (NEW METHOD)

Use of $ brackets and round bracket

① example:

```
CURRENT_WORKING_DIR=$(pwd)
```

② echo

```
date_time=$(date +"%D-%T")
echo ${date_time}
```

VIDEO 15 :-

READ ONLY & UNSET VARIABLE

```
name = zohaib
readonly name
echo "${name}"
```

UNSET :-

① name = (give no value to variable)

② unset name

VIDEO 16 :-

CONVERT A STRING TO UPPER & LOWER

```
String = "my name is zohaib"
echo "${string}"
echo "${string^}" # First letter would be capital.
echo "${String ^^}" # All letters would be capital.
echo "${string,}" # First letter would be small
echo "${string,,}" # All letters would be small.
```

LENGTH OF STRING VARIABLE

```
String = "My name is Gawrav"
echo "length of string variable is : ${#string}
```

VIDEO 17 :-

CONVERT STRING TO SUBSTRING (STRING MANIPULATION)

0 1 2 3 4 5 6 7

string = "a b c d e f g h c"

echo "${string:0}" # will print a

echo "${string:1}" # will print b

echo "${string:0:3}" # Start from 0 and go till 3

echo "${string: -5}" # with space after colon and go will in reverse order.

FROM STARTING

echo "${string#a*c}" # remove from a till c

echo "${string##a*c}" # remove with longest match with 'c'

FROM ENDING

echo "${string%%a*c}" # remove from ending

echo "${string%%%a*c}" # remove from ending with longest match

REPLACE :-

string "abcgouravabcxyz"

echo "${string/abc/xyz}" # replace abc with xyz

echo "${string//abc/xyz}" will replace abc with xyz in whole string

REMOVE :-

echo "${string/abc}" # will remove first abc

echo "${string //abc}" # remove abc every where from string"

VIDEO 18 :-

VARIABLE DEFAULT VALUE :-

① read -p "please enter name" name
echo "Hello ${name}"

② If no input detected on read:

read -p "please enter your name" name
name = ${name:-World} # default value of variable

③ yourname = ${unsetvariable-Manish}
echo $yourname

④ myname = "" # empty string
mytestname = ${myname-kali}
echo ${mytestname}

VIDEO 19 :-

CHECK COMMAND LINE VARIABLE PASSED OR NOT

: (NULL COMMAND EXIT STATUS ALWAYS SUCCESS)
name = "gaurav"
: ${name:?" please set variable value."}
echo "I m here."

If name value is not set then it will give you an error and exit from script.

VIDEO 20 :-

ARITHMETIC OPERATION :-

You need to use double round brackets for arithematic operations

```
a = 5
b = 6
echo "$((a+b))"        Addition
echo "$((5+b))"
echo "$((a-b))"        subtraction
echo "$((a*b))"        Multiplications.
echo "$((a/b))"        division
echo "$((a%b))"        reminder
echo "$((2**3))"       2^(3) = 2x2x2   Powa

((a++))
echo $a                # a=a+5
((a+=3))               # a=a+3
echo $a
```

VIDEO 21 :- FUNCTIONS

① with function

```
function install () {
    echo "Installation code 1"
    echo "Installation code 2"
}
install          # To call
```

(install — function name)

② without function

```
Configuration () {
    echo "Configuration Code 1"
    echo "Configuration Code 2"
}
Configuration    # To call
```

③ without round brackets

```
function deploy {
echo "config code 1"
echo "Config code 2"
}
deploy           # To call
```

If we call function prior to its execution then it will give an error.

VIDEO 22:

FUNCTION CALL WITHIN FUNCTION

```
function deploy () {
  # deploy code
  echo " deploy code "
}
function configuration () {
  # deploy config
  deploy
  echo " deploy config "
}
```

VIDEO 23

PASS ARGUMENT TO A FUNCTION :-

```
function install () {
  echo " Installing ${1} ${2} "
}
function depconfiguration () {
  echo " Config ${1}
}
install " " " webserver "
configuration " nginx "
```

To print function name

```
function install() {
    echo " Installing nginx "
    echo "${FUNCNAME}
```

echo ${0} will print script name

VIDEO 24 :-

If you want your variable only calls with in function then you need to use

"local myname = gaurav

VIDEO 25 :-

- If $? output is not zero then your last command was unsuccessful. (Output zero is success)
- Test command / test/check statement

```
# test a -eq 5
# 0 (success)
```

```
# test
# test a -eq 4
# 1. (unsuccessful)
```

VIDEO 26 :- (IF COMMAND)

If [COMMAND/CONDITION]
then
    COMMAND ONE
    COMMAND TWO
    COMMAND THREE
fi

VIDEO 27 :-

Use of square bracket

```
number = 5
if [ $number -eq 5 ]
then
    echo "Number is eq 5"
fi
```

VIDEO 28

Use double square brackets instead of single brackets

* read man test

VIDEO 29 :- (FILE CHECK)

To check file, directory, block, character device exists or not. along with read, write, execute permissions.

-b for block device.
-c for character device.
-d for directory check
-e for character special file
-e file exist

For help: man test

-r	read
-w	write
-x	execute

Example :-

```
#!/bin/bash
file_full_path="/home"
if [ -d ${file_full_path} ]
then
    echo "${file_full_path} is a dir"
fi
```

VIDEO 30 :- (NOT OPERATOR)

```
name="zohaib ahmed"
othername="shabbir ahmed"
if [[ ! ${name} == ${othername} ]]
then
    echo "true condition"
fi
```

VIDEO 31 :- (AND OPERATOR)

if first command remains successfull then it will go execute second command.

$ ls && echo "Hello World" &&

```
OS_Type=$(uname)
if [[ $OS_TYPE == "Linux" ]]
then
    if [[ $UID -eq 1000 ]]
    then
        echo "User is not root and OS is Linux"
    fi
fi
```

Instead of writing above program you may use AND operator.

```
#! /bin/bash
OS_TYPE=$(uname)
if [[ $OS_TYPE == "Linux" && $UID -eq 1000 ]]
then
    echo "User is not root & OS is Linux"
fi
```

VIDEO 32 :- (OR OPERATOR)

If first command gets fail then it will execute second command and it will be successful.

```
$ echo "Hello World" || date
Jul 6 09:14:25 PM PKT 2022.
```

VIDEO 33 :- (IF ELSE)

```
#! /bin/bash
name = "gaurav"
othername = "saurav sharma"
if [[ -n ${name} ]]
then
    echo "length of string is non zero"
else
    echo "length of string is zero"
fi
```

VIDEO 34 :- (NESTED IF ELSE)

```
#! /bin/bash
number = 10
if [[ ${number} -eq 10 ]]
then
    echo "Number is 10"
else
    if [[ ${number} -gt 10 ]]
    then echo "Number is greater than 10"
```

```
    else
        echo "Number is less than 10"
    fi
else
    echo "Number is less than or equal to 10"
fi
```

VIDEO 35:- (ELIF)

```
#! /bin/bash
if [[ ${number} -eq 10 ]]
then
    echo "Number is 10"
elif [[ ${number} -lt 10 ]]
then
    echo "Number is less than 10"
else
    echo "Number is greater than 10"
fi
```

VIDEO 36:- (CASE STATEMENT)

```
#! /bin/bash
action=${1}    # start, stop, restart, reload.
case ${action} in
start)
    echo "going to start"
    ;;
stop)
    echo "Going to STOP"
    ;;
```

```
reload)
    echo "Going to reload"
    ;;
restart)
    echo "Going to restart"
    ;;
*)
    echo "Please enter correct command line args."
esac.
```

VIDEO 37 :- Case statement with regex

```
#! /bin/bash
action=${1}
case ${action} in
    start | START)
    echo "Going to start"
    ;;
    stop | STOP)
    echo "Going to stop"
    ;;
esac
```

```
#! /bin/bash
action = ${1,,} — for lower case
case ${action} in
start)
 echo "Going to start"
 ;;
```

```
#! /bin/bash
read -p "enter Y or N" answer
case ${answer} in
 [Yy]) or [Yy][Ee][Ss])
      echo "You enter Yes"
      ;;
 [Nn])
      echo "You enter No"
      ;;
esac
```

VIDEO 38 (WHILE LOOP)

```
#! /bin/bash

while [[ $answer != "yes" ]]
do
   echo "You enter $answer"
done
```

* Above script will continue run until you stop.
. If you give input yes then you come out of while loop.

INFINITE LOOP :-

```
#! /bin/bash
while true
do echo "This is test"
done
```

COUNTING :-

```
#! /bin/bash
initNumber=1
while [[ ${initNumber} -le 10]]
do
   echo ${initNumber}
   ((initNumber++))
done
```

TABLE :-

```
#! /bin/bash
read -p "Please enter a Number" Num
initnumber=1
while [[ $initnumber -le 10]]
do
   echo $((initnumber * num))
   ((initNumber++))
done
```

To run your script with debug mode

$ bash -x while-loop.sh

VIDEO 39 :- (How to read file with While loop

```
#! /bin/bash
echo "My name is Zohaib" | while read line.
do echo "Printing line -> $line"
done
```

example

```
#! /bin/bash
cat /etc/passwd | while read line
do echo "$line"
done
```

example

```
#! /bin/bash
while read line
do echo "$line"
done < /etc/passwd
```

example

```
#! /bin/bash
while read line
do
    echo "$line"
    sleep 0.5 or 0.20
done < /etc/passwd
```

VIDEO 40 :- (UNTIL LOOP)

```
#! /bin/bash
read -p "please enter a number" number
initnumber=1
until [[ ${initnumber -eq 10]]
do
  echo $(( $initnumber * number ))
  ((initnumber++))
done
```

VIDEO 41 :- (FOR LOOP)

```
#! /bin/bash
for variableName in item1 item2 item3 item4
do
  echo "${variableName}"
done
```

example:

```
#! /bin/bash
for variableName in {1..10}
do echo $((variableName * number))
done
```

```
#! /bin/bash
for i in $(ls *.txt)
do
   echo "$i"
done
```

VIDEO 42 :-

```
#! bin/bash
echo "loop one"
for i in $*
do
  echo $i
done
```

RUN diffcommand.sh zohaib Ahmed shabbir

OUTPUT

zohaib
ahmeed
shabbir

DIFFERENCE B/W $* & $@

```
#! /bin/bash
for i in $@
do
   echo $@
done
```

RUN filename.sh zohaib ahmd shabb

outpot
zohaib
Ahmed
shabbir

WITH DOUBLE QUOTE :-

"$*" will consider all the input as one string
"$@" will treat consider every argument as individual

```
#! /bin/bash
for word in $(cat abc.txt)
do
   echo $word
   sleep 0.20
done
```

VIDEO 43:- (BREAK STATEMENT)

```
#! /bin/bash
initnumber=1
while [[ $initnumber -lt 10]]
do
    echo ${initnumber}
    if [[ $initnumber -eq 5 ]]
    then
        echo "Condition is true, number is $initnumber going
        #to break the loop
        break;
    fi
    ((initnumber++))
done
```

VIDEO 44:- (CONTINUE STATEMENT)

```
#! /bin/bash
initnumber=1
while [[ $initnumber -lt 10]]
do
    ((initnumber++))
    If [[ $initnumber -eq 5 ]]
    then
        continue
    fi
    echo ${initnumber}
done
```

OUTPUT:

```
2
3
4
6
7
8
9
10
```

"It will skip 5 and continue loop till condition met.

VIDEO 45 (NESTED LOOP)

```
#! /bin/bash
initNumber=1
while [[ $initNumber -lt 3 ]]
do
    for i in item1 item2 item3
    do
        echo "${initNumber - $i}"
    done
    ((initNumber++))
done
```

```
#! /bin/bash,
initNumber=
while [[ $initNumber -lt 3 ]]
do
    for i in item1 item2 item3
    do
        echo "${initNumber} -${i}"
        if [[ $i == item2 ]]
        then
            break;
        fi
    done
    ((initNumber++))
done
```

VIDEO 46 (SELECT)

```
#! /bin/bash
select OS in linux Window mac
do
    echo "You selected ${OS}"
done
```

```
#! /bin/bash
PS3="Please select OS;"  # To change #? while running
select OS in linux Windows Mac.
do
  case ${OS} in
  linux)
    echo "You selected linux"
    break
    ;;
  Windows)
    echo "You selected Windows"
    break
    ;;
  MAC)
    echo "You selected Mac"
    break
    ;;
  esac
done
```

Video 46 (Debug)

$bash -x scriptname.sh — for debugging

* Use "set -e" To exit from script when any command gives error.
* To enable debug mode for special lines like

```
#!/bin/bash
echo "My name is Zohaib"
var=5
echo "var is $var"
set -x          # To enable debug mode
testvar=10
echo "Testvar is $testvar"
set +x
```

* You may use set -xe /set +xe to debug and exist when error occur.