==========================

**Date : 02-Oct-2024**

**Topic : Shell Scripting**

==========================

=================

**Shell Scripting**

=================

=> Linux is free OS & open source

=> Multi User based os

=> Linux is secured

=> Linux is CLI based os

=> Linux is highly recommended for project related servers

Ex: Database, docker, sonar, jenkins, nexus, k8s....

================================

**Linux Architecture Components**

===============================

1) Applications / Commands

2) Shell

3) Kernel

4) Hardware components

===================

**What is shell ?**

===================

=> Shell acts as mediator between user and kernel.

=> Shell is responsible to process user given commands.

Note: when we execute a command, shell verify command syntax. If commad is valid then shell will convert that command into kernel understable format.

# check default shell of our linux vm

$ echo $SHELL

==========================

**What is Kernel in linux ?**

==========================

=> Kernel is heart of Linux OS

=> Kernel is a mediator between SHELL and Hardware components.

=> Kernel will get instructions from shell then kernel will convert that command into hardware understandable format.

# print kernel version

$ uname -r

# print linux distribution details

$ cat /etc/os-release

======================

What is scripting ?

======================

=> Scripting means set of commands we are keeping in a file for execution.

=> Scripting is used to automate our daily routine work.

=> For example, i want to execute below commands on daily basis

whoami

pwd

date

cal

ls -l

Note: instead of executing these commands one after other manually we can keep them inside a file and we can execute that file which is called as Scripting.

=> The process of executing script file using shell is called as shell scripting.

=> Shell scripting is used to automate our daily routine works in the project.

a) take backup

b) delete temp files

c) analyze log files

d) system health checks

=> shell script files will have .sh extension

Ex: backup.sh, log-analyzer.sh, health-checks.sh

============================

what is sha-bang in linux ?

============================

=> sha-bang is used to specify which shell we should use to process our script file.

Syntax:

#! /bin/bash

Note: Writing sha-bang is not mandatory but recommended.

============= 01 : Shell Script ===========

#! /bin/bash

echo "======== script start ======"

whoami

pwd

date

cal

echo "======== script end ========"

============= 02 : Shell Script ===========

#! /bin/bash

echo "Enter your name"

read FNAME

echo "Good Evening, $FNAME"

============= 03 : Shell Script ===========

#! /bin/bash

echo "Enter your first name"

read FNAME

echo "Enter your last name"

read LNAME

echo "Your Fullname : $FNAME + $LNAME"

====================

Scripting Concepts

====================

1) Variables

2) Operators

3) Conditional Statements (if-else)

4) Looping Statements (for, while)

5) Functions

6) Commandline Args

===========

Variables

===========

=> Variables are used to store the values

=> Variables will represent data in key-value format

a=20

b=40

name=ashok

gender=male

age=20

Note: We don't have data types in shell scripting.

=> We have 2 types of variables

1) System Variables / Environment Variables

2) User Defined Variables

=> The variables which are already defined and using by our system are called as System variables.

$ echo $SHELL

$ echo $USER

$ echo $PATH

Note: We can access all the environment variables using below command

$ env

=> The variables which we are creating based on our requirement are called as 'User Defined Variables'.

name = ashok

id = 101

age = 25

gender = male

Note : To access value of variable we will use below syntax

$ echo $VARIABLE\_NAME

# create variable using terminal

export course=devops

# get variable value

echo $course

# remove variable

unset variable\_name

Note: If we use export command in terminal for setting variables then those variables will be removed once we close our terminal. These are called as temporary variables.

====================================

How to set variables permanently ?

====================================

=> We will use ".bashrc" file to set variables permanently for the user.

$ ls -la

$ cat -n .bashrc

# open .bashrc

vi .bashrc

# add variables at end of the file

COURSE=devops

TRAINER=ashok

# apply .bashrc changes

source .bashrc

# Access variables

echo $COURSE

echo $TRAINER

Note: In linux machine, every user will contain their own .bashrc file.

==========================

Rules for Variables name

==========================

=> **Variable name should not start with digit**

Ex: 123name (invalid)

name123 (valid)

=> Variable name should not contain below 3 special symbols

Ex : - (hypen), @, #

**Note: It is recommended to use uppercase characters for variable names.**

name ==> NAME

age ==> AGE

============

Operators

============

=> Operators are used to perform some operation on the variables.

10 + 20 => 30

10 > 20 => false

40 == 40 => true

========================

Arithematic Operators

========================

Addition : +

Substraction : -

Multiplication : \*

Division : /

Modulas : %

Syntax to perform Arithematic Operations : $((VAR1 + VAR2))

================== 04 : Script (addition) ================

#! /bin/bash

echo "Enter First Number"

read FNUM

echo "Enter Second Number"

read SNUM

echo "Result : $((FNUM+SNUM))"

===================================

Relational/Comparision Operators

===================================

Equals :: == or eq

Not Equals :: !=

Greater Than :: > or gt > ge>=

Less Than :: < or lt or le <=

==============================================

Date : 03-Oct-2024

Topic : Conditional Stmts & Looping stmts

==============================================

=> Conditional statements are used to execute commands based on condition.

Ex :

read user age

if age is above 18 years then print msg as "eligible for vote"

if age is below 18 years then print msg as "not-elgible for vote"

=> To implement conditional stmts we will use "if-elif-else"

Syntax :

if [ condition-1 ]; then

// stmts or commands

elif [ condition-2 ]; then

// stmts or commands

else

// stmts

================== 05 : Script ( if-else) ================

#! /bin/bash

echo "Enter age"

read AGE

if [ $AGE -ge 18 ]; then

echo "Eligible For Vote"

else

echo "Not Eligible for Vote"

fi

===========================================================

Requirement : Take a number from user and check weather it is positive or negative or zero

===========================================================

#! /bin/bash

echo "Enter number"

read A

if [ $A -gt 0 ]; then

echo "Positive num"

elif [ $A -lt 0 ]; then

echo "Negative Num"

else

echo "It is zero"

fi

===========================================================

Requirement : Take a number from user and check given number is even or odd

===========================================================

#! /bin/bash

echo "Enter number"

read A

if (( $A % 2 == 0 )); then

echo "$A is even"

else

echo "$A is odd"

fi

====================

**Looping Statements**

====================

=> Loops are used to execute statements multiple times.

=> In scripting we can use 2 types of loops

1) **Range based loop (Ex: for)**

2) **Conditional based loop (Ex: while)**

---------------------

for loop syntax :

---------------------

for(( initialization; condition; modification ))

do

//stmts

done

================================================

**For loop example - Print Numbers from 1 to 10**

================================================

#! /bin/bash

for((i=1; i<=10; i++))

do

echo $i

done

================================================

**For loop example - Print Numbers from 10 to 1**

================================================

#! /bin/bash

for((i=10; i>=1; i--))

do

echo $i

done

===========================================================

Requirement: Write shell script to print table of given number

input = 2

output :

2 \* 1 = 2

2 \* 2 = 4

..

2 \* 10 = 20

=================================

Date : 04-Oct-2024

Topic : Looping stmts

=================================

=============

While Loop

=============

=> While loop is used to execute statements until condition is true.

syntax:

while [ condition ]

do

// stmts

done

Multiplication table using while loop

#! /bin/bash

echo " enter a number : "

read N

i=1

while [ $i -le 10 ]

do

echo " $N \* $i = $(( N\*i )) "

let i++

done

==================================================================

print nums from 1 to 10 using while loop

------------------------------------------

#! /bin/bash

N=1

while [ $N -le 10 ]

do

echo $N

let N++

done

============================

Print Numbers from 10 to 1

============================

#! /bin/bash

N=10

while [ $N -ge 1]

do

echo $N

let N--

done

======================

**Functions / Methods**

======================

=> Functions are used to perform some action / task.

=> Using functions we can divide big task into multiple small tasks.

=> Using functions we can divide our work logically.

=> Functions are re-usable.

-------

syntax

-------

# creating function

function welcome() {

// body

}

# calling function to execute

welcome

====================== shell script with Function ===============

#! /bin/bash

function welcome(){

echo "Welcome to ashokit"

echo "welcome to devops"

echo "welcome to aws"

}

welcome

======================================================

how to call one sh file function in another sh file

======================================================

------------------------01-script.sh---------------

#!/bin/bash

my\_function() {

echo "Hello from my\_function!"

}

------------------------02-script.sh-----------------

#! /bin/bash

source ./01-script.sh

my\_function

-----------------------------------------------------

========================

Command Line Arguments

========================

=> cmd args are used to supply values to script file at the time of script execution.

ex:

sh task.sh ashokit devops aws linux

=> We can read cmd args in script like below

$# : Total no.of cmd args

$1 : read first cmd arg

$2 : read second cmd arg

$3 : read third cmd arg

$\* : Read all cmd args

-------------------------------------

#! /bin/bash

echo "Total args : $#"

echo "======================="

echo "First cmd args: $1"

echo "Second cmd args: $2"

echo "======================="

echo "All cmd args: $\*"

------------------------------------

=====================================================================

Requirement: Write shell script to perform sum of two numbers using cmd args

=====================================================================

#! /bin/bash

echo "Result : $(($1+$2))"

$((1+2))

========================= Assignments ==============================

1) Write a shell script to check given number is prime or not

2) Write a shell script to check given text is palindrome or not

Ex: madam, liril, ashok

3) Write a shell script to take backup of current working directory

4) Write a shell script to check with given name file is available or not, if not available then create that file.

Ex: abc.txt

#! /bin/bash

=====================

Date : 05-Oct-2024

Topic : CRON JOBS

=====================

====================

What is Scheduling

====================

shell script file : system-health-check.sh

Requriement :: Everyday @9:00 AM we have to run above shell script file.

=> Instead of running that file manually, we can use scheduling.

=> Scheduling means configuring the tasks to be executed automatically.

=> In linux, we will use CRON to schedule jobs/scripts execution.

=> CRON is an utility in linux to schedule jobs execution.

=> In real-time we will use several jobs on daily/weekly/monthly/yearly basis to automate our work.

- Delete Temp files

- Take backup of files

- System health check

================

**CRON JOB Syntax**

================

Syntax : \* \* \* \* \* <script-file-name>

=> First \* will represent minutes (0 - 59)

=> Second \* will represent hour ( 0 - 23 )

=> Third \* will represent day of month ( 1 - 31 )

=> Fourth \* will represent month of year ( 1 - 12 )

=> Fifth \* will represent day of week ( 0 - 6 / sun - sat)

======================

Sample CRON Schedules

======================

# Run for every 15 mins

\*/15 \* \* \* \* backup.sh

# Run everyday @5:00 AM

0 5 \* \* \* backup.sh

# Run everyday @5:00 PM

0 17 \* \* \* backup.sh

# Run every month on first day @9:00 AM

0 9 1 \* \* backup.sh

# Run job everyday @4:15 PM Monday - Friday

15 16 \* \* 1-5 backup.sh

Note: We can write cron expression using below website

####### URL : https://crontab.guru/ ######

**======================================**

**Where to configure cronjob in linux** ?

======================================

-> crontab file is used to configure cronjobs in linux

# open crontab file

crontab -e

# display cronjobs schedules

crontab -l

# remove crontab file

crontab -r

====================

**CRONJOB Practicals**

====================

Que - Why we launch Ubuntu vm for cron jobs can we use different vm ?

1) Launch Linux machine with UBUNTU AMI

2) Connect with Linux VM using SSH client

3) Create shell script file and keep below content

$ vi task.sh

touch /home/ubuntu/f1.txt

touch /home/ubuntu/f2.txt

**Note : -cron job will responsible to run this file.**

4) Provide execute permission for script file

$ chmod +x task.sh

5) Note:-Open crontab file and configure job schedule

$ crontab -e

Note: Add below job schedule info

\*/1 \* \* \* \* /bin/bash /home/ubuntu/task.sh

6) Save and close the crontab file (ctrl + x + y + enter)

7) After 1 minute check files got created or not.

$ ls -l

8) To remove crontab file we can use below command

$ crontab -r

=========

Summary

=========

1) Linux Architecture

2) What is shell

3) What is kernel

4) What is shell scripting & why ?

5) sha-bang

6) Variables (env & user-defined)

7) .bashrc file

8) cmd args

9) Operators

10) Conditional Statements

11) Looping Statements

12) Functions

13) Cron Jobs