Date 06/02
Page

* Shell Scripting *

- Shell Script :- shell scripting is used to automate repetative system task. Put your linux commiands in one file.

1) vim filename.sh :- to create a shell script file.
2) ./shellscriptfilename :- to execute schell script.

* Crithub.com :- Store the source code. Work as repository.

3) git clone <repository URL> :- to create a copy of an existing Crit repository in a new dir in our local.

- #!/bin/bash :- This is also called as shebang. Interpretor this indicate this is not normal file it is shell script file.

ex:- 1)
```
echo "value of a"
read a  → it will ask to enter value for (a)
echo "Hello value of a is $a"
                    ↳ this is variable,
```
if we want to call var we need to insert $.
$ → dollar is used to call the variables.

2)
```
Name = Young-Minds
readonly Name
Name = devops
echo "my name is : $NAME"
```
This script is to make variable read only, we cannot set the value of Name variable again.

1) Local variable :-

ex :- func () {
Name = Ram
}

2) Enivormental / Global variable :-

ex :- 1. PATH echo $PATH → variable for whole
2. HOME echo $ Home, Operating System
/root

3) export VariableName = value :- to set a variable.
ex :- export JDKHOME = /bin/jdk

3) Shell Variable :-

variable name ÷ variable value.

a = 10
b = 20
class = young minds

- Variable :- variable work is store the value when we call that variable the value we get.

ex :- a=10
echo = $a
Output → 10

* Arithmetic Operators :- There are various operator in this like :

addition :- 'expr $a + $b'
subtraction :- 'expr $a - $b'
multiplication :- 'expr $a * $b'
Division :- 'expr $a / $b'
Modulus :- 'expr $a % $b'

Relation Operators :- Equal too, Not equal too, greater than, less than

a = 10 b = 20

[$a -eq $b] is not true because both are not equal

[$a -ne $b] is true — " — equal

[$a -gt $b] a greater than b or not

[$a -lt $b] check if a is less than b or not.

Boolean Operators :- The true or false

0 = False 1 = True

[! false] is True → vise versa

[! True] is false

[$a -lt 20 -o $b -gt 100]

1 -o 0

1 → True

[$a -lt 20 -a $b -gt 100]

1 -a 0

0 → False

if both 1 then only 1

a	b	a AND b	a OR b
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

OR → if any value is 1 then 1

String Operators :- Assume a = abc & b = egf

[$a = $b] is not true because a is not equal to b

[$a != $b] is true both are not same

[-z $a] is not true because length is not zero

[-n $a] true because it is non-zero

[$a] true " " not empty string.

* File Test Operators :-

-f $ filename :- check if the file is available or not to in which location you are.

-d $ filename :- check if file is a directory if yes then condition become True.

* Exit Status *

** 4) echo $? :- if any previous given command is True then the value will come 0 and non-zero value will come if previous command is wrong.

ex:-

```
ls
echo $?
0 → command right
ddate
echo $?
126 → command wrong
```

5) $ 0 :- self shell script
6) $ 1 :- 1st cmdline argument
7) $ 2 :- 2nd " "
8) $ a :- list of " "
9) $ # :- no. of " "

* Arithmethic ex :-

①
```
a=10
b=20
val1='Expr $a + $b'
echo "a+b : $val1"
```

② if [$a = $b]
then
echo "The valii of a and b is equal"
fi

③ if [$a = $b]
then
echo "The value of a and b is equal"
else
echo "The value of a and b is not equ"
fi

④ if [$a = $b]
then
echo "The value of a and b is equal"
elif
then [$a -gt $b]
echo "The vale of a is greater than b"

elif [$a -lt $b]
then
echo "The value of a is less than b"
else
echo "None of condition met"
fi

⑤ echo "Enter the value of a"
read a
echo "Enter the value of b"
read b
if [$a -lt 100 -a $b -gt 15]
then
echo "$a -lt 100 -a $b -gt 15 : returns True"
fi

⑥ for i in 12345 → if we want to take more no. the {1..50}
do
echo "Looping ... number $i"
~~done~~ sleep 2 → it will take 2 sec brake
done

⑦ ~~for i in~~
~~do~~ i=1
for day in Mon, Tue, Wed Thur fri
do
echo "weekday $((i++)) : $day" (i+1 increment)
~~done~~ sleep 3
done

⑧ a=0
~~do~~ while [$a -lt 10]
do
echo $a
a=`expr $a + 1`
done

⑨ Hello () {
echo "Hello world"
}
Hello → calling our function

10) number_one () {
echo "First function"
number_two
}
number_two () {
echo "Second Function"
}
number_one ()

The types of shell *

1) Bourn shell (sh)
2) C shell (csh)
3) korn shell (ksh)
4) Bourne again shell (bash)

Bourn shell (sh) :-

Orignal UNIX shell. Faster and more preffered. Lacks of feature like recall previous cmd and built-in arithmetic & logical expression handling.

C shell (csh) :-

It include helpful programing feature like built-in arithemetic & C-like expression syntax.

Korn shell (ksh) :-

Developed by David korn. It is superset of Bourne Shell. It includes built in feature likes :-
C-like arrays,
function, String Manipulation
faster than C-shell.

Bourne Again Shell (bash) :-

Developed by Freeware Soft and licensed under GNU. Comptible to bourne shell. Include feature from Korn & Born. Free to use and open source.