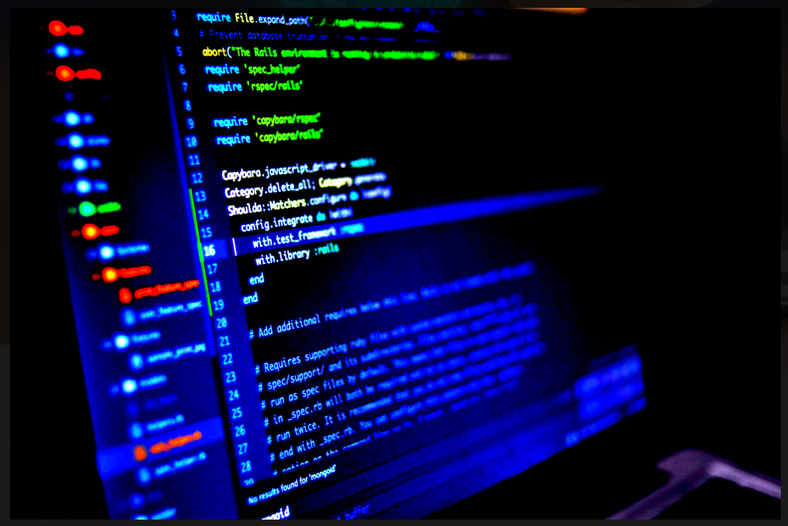
MPRASHANT

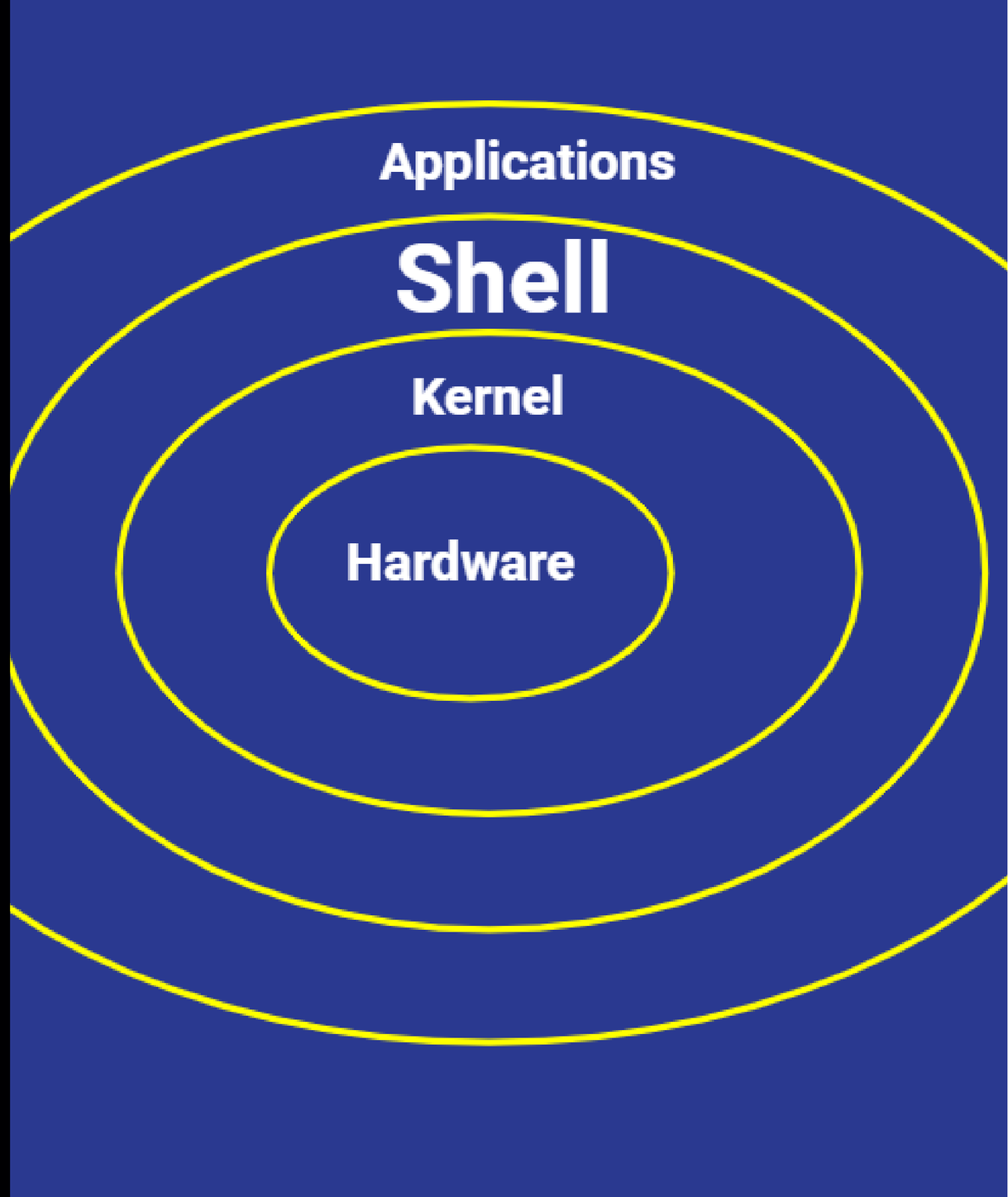
SHELL SCRIPTING



MPRASHANT

WHAT IS LINUX SHELL?

A shell provide an environment to a user to execute commands and interact with kernel.



There are different types of shell

- bash
- sh
- ksh
- tsh
- fish
- zsh

MPRASHANT

WHAT IS MY SHELL TYPE?

You can check using

```
echo $0
```

MPRASHANT

WHAT IS SHELL SCRIPTING?

- Shell script consist of set of commands to perform a task.
- All the commands execute sequentially.
- Some task like file manipulation, program execution, user interaction, automation of task etc can be done

==========
Types of Shells
==========

1) Bourne Shell

2) Bash Shell

3) Korn Shell

4) CShell

5) TShell

Display list of shells availabe in Linux Machine

$ cat /etc/shells

display default shell of our linux machine

$ echo $SHELL

MPRASHANT

FIRST BASIC SCRIPT...

```
#!/bin/bash
echo "Hello World!"
```

MPRASHANT

WHAT IS SHEBANG?

```
#!/bin/bash
```

```
================================
What is SHA-BANG in shell scripting ?
================================
=> sha-bang is used to specify which shell we want to use to process our script file.
Syntax:
#! /bin/bash
```

MPRASHANT

SENDING OUTPUT TO TERMINAL..

```
echo "Hello World!"
```

MPRASHANT

HOW TO RUN A SCRIPT..

- Make sure script has execute permission `rwx`

- Run using
  - `./script.sh`
  - `/path/script.sh`
  - `bash script.sh`

- Ctrl+C to terminate
- Ctrl+z to stop

MPRASHANT

COMMENTS

Using #

```
#This is a comment
```

Multi-line comment

```
<<comment
...
your comment here
...
comment
```

MPRASHANT

WHAT ARE VARIABLES?

```
VAR_NAME=value

VAR_NAME=$(hostname)

echo $VAR_NAME
```

==========Variables==========

=> Variables are used to store the values

id=101
name=ashok
gender=male

=> Variables will represent data in key-value pair

=> Variables are divided into 2 types

1) Environment Variables / System Variables
2) User Defined variables

=> The variables which are using by our System are called as Environment / System variables.

> echo $HOSTNAME
> echo $NAME
> echo $SHELL

=> The variables which we are creating based on our requirement those are called as user-defined variables

course=devops
duration=4 months

==========
Variable Name Rules
==========

=> Variable Name should n't start with digit

=> We can't use special symbols like @, # etc...

=> Recommended to use Uppercase characters for variable name

```
#!/bin/bash

#Script to show how to use variables

a=10
name="Prashant"
age=28

echo "My name is $name and age is $age"

name="Paul"

echo "My name is $name"

#Var to store the output of a command
HOSTNAME=$(hostname)
echo "Name of this machine is $HOSTNAME"
```

MPRASHANT

CONSTANT VARIABLE?

Once you defined a variable and don't wanna change it until end of the script.

```
readonly var_name="Hi"
```

```
#!/bin/bash

#Constant variable
readonly COLLEGE="METRO"

echo "My college name is $COLLEGE"

COLLEGE="TEST"
```

```
[paul@redhat01 myscripts]$ bash 04_constant_var.sh
My college name is METRO
[paul@redhat01 myscripts]$ vi 04_constant_var.sh
[paul@redhat01 myscripts]$ bash 04_constant_var.sh
My college name is METRO
04_constant_var.sh: line 8: COLLEGE: readonly variable
```

MPRASHANT

ARRAYS

How to define an array?

```
myArray=( 1 2 Hello "Hey man")
```

How to get values from an array?

```
echo "${myArray[0]}"
echo "${myArray[1]}"
```

```
#!/bin/bash

#Array
myArray=( 1 20 30.5 Hello "Hey Buddy!" )

echo "All the values in array are ${myArray[*]}"
echo "Value in 3rd index ${myArray[3]}"

#How to find no. of values in an array
echo "No. of values, length of an array is ${#myArray[*]}"

echo "Values from index 2-3 ${myArray[*]:2:2}"
```

```
[paul@redhat01 myscripts]$ vi 05_arrays.sh
[paul@redhat01 myscripts]$ bash 05_arrays.sh
All the values in array are 1 20 30.5 Hello Hey Buddy!
Value in 3rd index Hello
No. of values, length of an array is 5
Values from index 2-3 30.5 Hello
```

ARRAYS

How to get length of array?

```
echo "${#myArray[*]}"
```

How to get specific values?

```
echo "${myArray[*]:1}"
```

```
echo "${myArray[*]:1:2}"
```

MPRASHANT

ARRAYS

How to update an array?

```
myArray+=( 5 6 8 )
```

```
#Updating our array with new values
myArray+=( New 30 40 )

echo "Values of new array are ${myArray[*]}"
```

```
Values of new array are 1 20 30.5 Hello Hey Buddy! New 30 40
```

MPRASHANT

ARRAYS KEY-VALUE

```
declare -A myArray
myArray=( [name]=Paul [age]=20 )

echo "${myArray[name]}"
```

```
#!/bin/bash

#How to store the key values pairs

declare -A myArray
myArray=( [name]=Prashant [age]=28 [city]=Paris )

echo "Name is ${myArray[name]}"
echo "Age is ${myArray[city]}"
```

```
[paul@redhat01 myscripts]$ bash 06_key-value.sh
Name is Prashant
Age is Paris
```

String Operations

MPRASHANT

STRING OPERATIONS

```
myVar="Hello World!"

length=${#myVar}

upper=${x^^}

lower=${y,,}

replace=${myVar/World/Buddy}

slice=${myVar:6:11}
```

```
myVar="Hey Buddy, How are you?"

myVarLength=${#myVar}
echo "Length of the myVar is $myVarLength"

echo "Upper case is ----- ${myVar^^}"
echo "Lower case is ----- ${myVar,,}"

#To replace a string
newVar=${myVar/Buddy/Paul}
echo "New Var is ---- $newVar"

#To slice a string
echo "After slice ${myVar:4:5}"
```

```
[paul@redhat01 myscripts]$ vi 07_string_ops.sh
[paul@redhat01 myscripts]$ bash 07_string_ops.sh
Length of the myVar is 23
Upper case is ----- HEY BUDDY, HOW ARE YOU?
Lower case is ----- hey buddy, how are you?
New Var is ---- Hey Paul, How are you?
After slice Buddy
```

User Interaction

MPRASHANT

```
#!/bin/bash

echo "What is your name? - "
read name
echo "Your name is $name"
```

```
[paul@redhat01 myscripts]$ bash 08_user_int.sh
What is your name? -
Paul
Your name is Paul
```

```
#!/bin/bash

read -p "What is your name? " name
echo "Your name is $name"
```

```
[paul@redhat01 myscripts]$ bash 08_user_int.sh
What is your name? Paul
Your name is Paul
```

TAKING INPUT FROM USER...

`read <var_name>`

`read -p "Your name" NAME`

```
#!/bin/bash

read name
echo "Your name is $name"
```

```
[paul@redhat01 myscripts]$ bash 08_user_int.sh
Paul
Your name is Paul
[paul@redhat01 myscripts]$ bash 08_user_int.sh
Prashant
Your name is Prashant
```

Arithmetic Operations

MPRASHANT

HOW TO USE EXPRESSIONS

Using let command

```
let a++
let a=5*10
```

```
((a++))
((a=5*10))
```

```
#!/bin/bash

#Maths Calculation
x=10
y=2

let mul=$x*$y
echo "$mul"

let sum=$x+$y
echo "$sum"

echo "substraction is $(($x-$y))"
```

```
[paul@redhat01 myscripts]$ bash 09_airth_ops.sh
20
12
substraction is 8
```

Conditional Statement

MPRASHANT

IF-ELSE

```
if [ $marks -gt 40 ]
then
        echo "You are PASS"
else
        echo "You are FAIL"
fi
```

```
#!/bin/bash

read -p "Enter your marks: " marks

if [[ $marks -gt 40 ]]
then
        echo "You are PASS"
else
        echo "You are FAIL!!!!!!!!!!!!"
```

```
[paul@redhat01 myscripts]$ bash 10_if-else.sh
Enter your marks: 30
You are FAIL!!!!!!!!!!!!
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ bash 10_if-else.sh
Enter your marks: 41
You are PASS
[paul@redhat01 myscripts]$
```

MPRASHANT

OPERATORS

Equal	-eq / ==
Greaterthanorequalto	-ge
Lessthanorequalto	-le
Not Equal	-ne / !=
Greater Than	-gt
Less Than	-lt

MPRASHANT

ELIF

```
if [ $marks -ge 80 ]
then
        echo "First Division""
elif [ $marks -ge 60 ]
then
        echo "Second Division"
else
        echo "Fail""
fi
```

```bash
#!/bin/bash

read -p "Enter your marks: " marks

if [[ $marks -ge 80 ]]
then
        echo "1st Division"
elif [[ $marks -ge 60 ]]
then
        echo "2nd Division"
elif [[ $marks -ge 40 ]]
then
        echo "3rd Division"
else
        echo "You are FAIL!!!!!!!!!!!!"
fi
```

```
[paul@redhat01 myscripts]$ vi 11_elif-demo.sh
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ bash 11_elif-demo.sh
Enter your marks: 90
1st Division
[paul@redhat01 myscripts]$ bash 11_elif-demo.sh
Enter your marks: 70
2nd Division
[paul@redhat01 myscripts]$ bash 11_elif-demo.sh
Enter your marks: 50
3rd Division
[paul@redhat01 myscripts]$ bash 11_elif-demo.sh
Enter your marks: 30
You are FAIL!!!!!!!!!!!!
```

MPRASHANT

CASE

```
echo "Hey choose an option"
echo "a = To see the current date"
echo "b = list all the files in current dir"

read choice

case $choice in
        a) date;;
        b) ls;;
        *) echo "Non a valid input"
esac
```

```
echo "Provide an option"
echo "a for print date"
echo "b for list of scripts"
echo "c to check the current location"

read choice

case $choice in
        a)date;;
        b)ls;;
        c)pwd;;
        *)echo "Please provide a valid value"
esac
```

```
[paul@redhat01 myscripts]$ bash 12_case_demo.sh
Provide an option
a for print date
b for list of scripts
c to check the current location
a
Wednesday 26 July 2023 09:14:21 AM EEST
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ bash 12_case_demo.sh
Provide an option
a for print date
b for list of scripts
c to check the current location
b
01_basic.sh     04_constant_var.sh  07_string_ops.sh
02_comments.sh  05_arrays.sh        08_user_int.sh
03_vardemo.sh   06_key-value.sh     09_airth_ops.sh
```

```
c
/home/paul/myscripts
```

```
#!/bin/bash

echo "Provide an option"
echo "a for print date"
echo "b for list of scripts"
echo "c to check the current location"

read choice

case $choice in
        a)
                echo "Today's date is:"
                date
                echo "Ending..."
                ;;
        b)ls;;
        c)pwd;;
        *)echo "Please provide a valid value"
esac
```

```
[paul@redhat01 myscripts]$ bash 12_case_demo.sh
Provide an option
a for print date
b for list of scripts
c to check the current location
a
Today's date is:
Wednesday 26 July 2023 09:16:31 AM EEST
Ending...
```

Logical Operators
&&, ||, !

MPRASHANT

LOGICAL OPERATORS

```
condition1 && condition2
```

If both conditions are true then true else false

```
condition1 || condition2
```

If any of the condition is true then true

```
#!/bin/bash

#AND Operator

read -p "What is your age? " age
read -p "Your country: " country

if [[ $age -ge 18 ]] && [[ $country -eq "India" ]]
then
        echo "You can vote"
else
        echo "You can't vote"
fi
```

```
[paul@redhat01 myscripts]$ bash 13_logical-ops.sh
What is your age? 18
Your country: India
You can vote
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ bash 13_logical-ops.sh
What is your age? 20
Your country: Nepal
You can vote
```

যখনি আমরা যেকোন comparison numerically করব তখন কিন্তু আমাদের -eq use করলে হবে কিন্তু যদি আমরা string কেও comparison করতে চায় সেক্ষেত্রে == use করতে হবে,

```
#!/bin/bash

#AND Operator

read -p "What is your age? " age
read -p "Your country: " country

if [[ $age -ge 18 ]] && [[ $country == "India" ]]
then
        echo "You can vote"
else
        echo "You can't vote"
fi
```

```
[paul@redhat01 myscripts]$ bash 13_logical-ops.sh
What is your age? 18
Your country: India
You can vote
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ bash 13_logical-ops.sh
What is your age? 18
Your country: Nepal
You can't vote
[paul@redhat01 myscripts]$ bash 13_logical-ops.sh
What is your age? 15
Your country: India
You can't vote
```

```bash
#!/bin/bash

#AND Operator

read -p "What is your age? " age
read -p "Your country: " country

if [[ $age -ge 18 ]] || [[ $country == "India" ]]
then
        echo "You can vote"
else
        echo "You can't vote"
fi
```

```
[paul@redhat01 myscripts]$ vi 13_logical-ops.sh
[paul@redhat01 myscripts]$ bash 13_logical-ops.sh
What is your age? 20
Your country: Nepal
You can vote
[paul@redhat01 myscripts]$ bash 13_logical-ops.sh
What is your age? 15
Your country: India
You can vote
[paul@redhat01 myscripts]$ bash 13_logical-ops.sh
What is your age? 20
Your country: India
You can vote
```

MPRASHANT

LOGICAL OPERATORS

```
condition1 && condition2 || condition3
```

Execute condition2 only when 1 is true else execute condtion3

```
#!/bin/bash

#cond1 && cond2 || cond3

age=18

[[ $age -ge 18 ]] && echo "Adult" || echo "Minor"
```

```
[paul@redhat01 myscripts]$ bash 14_ternary-ops.sh
Adult
```

```
#!/bin/bash

#cond1 && cond2 || cond3

age=15

[[ $age -ge 18 ]] && echo "Adult" || echo "Minor"
```

```
[paul@redhat01 myscripts]$ bash 14_ternary-ops.sh
Minor
```

Loops

MPRASHANT

FOR LOOP

```
for i in 1 2 3 4 5
do
        echo "Number is $i"
done
```

Other ways to write For loop

```
for j in Raju Sham Baburao
for p in {1..20}
```

```
#!/bin/bash

for i in 1 2 3 4 5 6 7 8 9 10
do
        echo "Number is $i"
done
```

```
[paul@redhat01 myscripts]$ bash 15_forloop1.sh
Number is 1
Number is 2
Number is 3
Number is 4
Number is 5
Number is 6
Number is 7
Number is 8
Number is 9
Number is 10
```

```
#/bin/bash
for loop with infinite
for (( ;; ))
do
        echo "Hi buddy"
        sleep 2s
done
```

```
Hi buddy
Hi buddy
```

```
#!/bin/bash

for i in 1 2 3 4 5 6 7 8 9 10
do
        echo "Number is $i"
done

for name in Raju Sham Baburao
do
        echo "Name is $name"
done
```

```
[paul@redhat01 myscripts]$ bash 15_forloop1.sh
Number is 1
Number is 2
Number is 3
Number is 4
Number is 5
Number is 6
Number is 7
Number is 8
Number is 9
Number is 10
Name is Raju
Name is Sham
Name is Baburao
```

MPRASHANT

ITERATE VALUES FROM FILE..

```
items="/home/paul/file.txt"

for item in $(cat $items)
do
        echo $item
done
```

```
#!/bin/bash

#Getting values from a file names.txt

FILE="/home/paul/myscripts/names.txt"

for name in $(cat $FILE)
do
        echo "Name is $name"
done
```

```
[paul@redhat01 myscripts]$ bash 16_for_with_file.sh
Name is Raju
Name is Sham
Name is Baburao
Name is Paul
Name is Alex
```

```
#!/bin/bash

myArray=( 1 2 3 Hello Hi )

length=${#myArray[*]}

for (( i=0;i<$length;i++ ))
do
        echo "Value of array is ${myArray[$i]}"
done
```

```
[paul@redhat01 myscripts]$ bash 17_for_with_array.sh
Value of array is 1
Value of array is 2
Value of array is 3
Value of array is Hello
Value of array is Hi
```

MPRASHANT

WHILE LOOP

```
count=0
num=10

while [ $count -le $num ]
do
        echo "Numbers are $count"
        let count++

done
```

```
#!/bin/bash

count=0
num=10

while [[ $count -le $num ]]
do
        echo "Value of count var is $count"
        let count++
done
```

```
[paul@redhat01 myscripts]$ bash 18_while_demo.sh
Value of count var is 0
Value of count var is 1
Value of count var is 2
Value of count var is 3
Value of count var is 4
Value of count var is 5
Value of count var is 6
Value of count var is 7
Value of count var is 8
Value of count var is 9
Value of count var is 10
```

MPRASHANT

UNTIL LOOP..

```
a=10

until [ $a -eq 1 ]
do
    echo $a
    a=`expr $a - 1`
done
```

```
#!/bin/bash

a=10

until [[ $a -eq 1 ]]
do
        echo "Value of a is $a"
        let a--
done
```

```
[paul@redhat01 myscripts]$ bash 19_until_loop.sh
Value of a is 10
Value of a is 9
Value of a is 8
Value of a is 7
Value of a is 6
Value of a is 5
Value of a is 4
Value of a is 3
Value of a is 2
```

MPRASHANT

INFINITE LOOP..

```
while true
do
    echo "Hi'
    sleep 2s
done
```

```
#!/bin/bash

#infinite loop

while true
do
        echo "Hi buddy"
        sleep 2s
done
```

```
Hi buddy
Hi buddy
Hi buddy
Hi buddy
Hi buddy
Hi buddy
Hi buddy
Hi buddy
Hi buddy
```

MPRASHANT

WHILE LOOP

To read content from a file

```
while read myVar
do
        echo $myVar

done < file_name
```

```
#!/bin/bash

while read myvar
do
        echo "Value from file is $myvar"
done < names.txt
```

```
[paul@redhat01 myscripts]$ bash 22_while_with_file.sh
Value from file is Raju
Value from file is Sham
Value from file is Baburao
Value from file is Paul
Value from file is Alex
Value from file is John
Value from file is Salman
```

MPRASHANT

WHILE LOOP

To read content from a csv file

```
while IFS="," read f1 f2 f3
do
        echo $f1
        echo $f2
        echo $f3
done < file_name.csv
```

```
#!/bin/bash

while IFS="," id name age
do
        echo "Id is $id"
        echo "name is $name"
        echo "age is $age"
done < test.csv
```

Internal Field Seperator

```
Id is id
name is name
age is age
Id is 01
name is paul
age is 20
Id is 02
name is alex
age is 30
Id is 03
name is raju
age is 40
```

```
#!/bin/bash

while IFS="," read id name age
do
        echo "Id is $id"
        #echo "name is $name"
        #echo "age is $age"
done < test.csv
```

```
Id is id
Id is 01
Id is 02
Id is 03
```

MPRASHANT

WHILE LOOP

To read content from a csv file

Use NR!=1

```
cat myfile.csv | awk '!NR=1 {print}'
| while IFS="," read f1 f2 f3
```

```
#!/bin/bash

cat test.csv | awk 'NR!=1 {print}' | while IFS="," read id name age
do
        echo "Id is $id"
        #echo "name is $name"
        #echo "age is $age"
done
```

```
[paul@redhat01 myscripts]$ bash 23_while_with_csv.sh
Id is 01
Id is 02
Id is 03
[paul@redhat01 myscripts]$ cat test.csv
id,name,age
01,paul,20
02,alex,30
03,raju,40
[paul@redhat01 myscripts]$ cat test.csv | awk 'NR!=1 {print}'
01,paul,20
02,alex,30
03,raju,40
```

Functions

MPRASHANT

WHAT ARE FUNCTIONS?

- Block of code which perform some task and run when it is called.
- Can be reuse many times in our program which lessen our lines of code.
- We can pass arguments to the method

MPRASHANT

HOW TO MAKE FUNCTIONS?

```
function myfun {
    echo "Hi"
}

myfun() {
    echo "Hello"
}
--------------------------------

To call the function
myfun
```

MPRASHANT

HOW TO USE ARGUMENTS IN FUNCTIONS?

```
addition() {
        local num1=$1
        local num2=$2
        let sum=$num1+$num2
 echo “Sum of $num1 and $num2 is $sum”
}

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

myfun 12 13
```

```bash
#!/bin/bash

#To make function
function welcomeNote {
        echo "----------------"
        echo "Welcome"
        echo "----------------"
}

#To call our function
welcomeNote
welcomeNote
welcomeNote
```

```
----------------
Welcome
----------------
----------------
Welcome
----------------
----------------
Welcome
----------------
```

```bash
#!/bin/bash

#To make function
welcomeNote() {
        echo "----------------"
        echo "Welcome"
        echo "----------------"
}

#To call our function
welcomeNote
welcomeNote
welcomeNote
```

```
----------------
Welcome
----------------
----------------
Welcome
----------------
----------------
Welcome
----------------
```

```bash
#!/bin/bash

function welcomeNote {
        echo "----------------"
        echo "Welcome $1"
        echo "----------------"
}

welcomeNote Raju
welcomeNote Sham
```

```
----------------
Welcome Raju
----------------
----------------
Welcome Sham
----------------
```

```bash
#!/bin/bash

function welcomeNote {
        echo "----------------"
        echo "Welcome $1"
        echo "Age is $2"
        echo "----------------"
}

welcomeNote Raju 20
welcomeNote Sham 30
```

```
----------------
Welcome Raju
Age is 20
----------------
----------------
Welcome Sham
Age is 30
----------------
```

Arguments Passing

MPRASHANT

ARGUMENTS IN SCRIPT...

```
#myscript.sh arg1 arg2..

How to access these arguments inside
our script?

To get no. of arguments     : $#
To display all arguments    : $@
To use or display a argument: $1 $2 ..
```

==========
Command Line Arguments
==========

=> CMD Args are used to supply the values while running script file

ex: sh task.sh 10 20

=> We can read command line args in script file like below

$# ===> No.of args

$0 ===> Script file name

$1 ===> First cmd args

$2 ===> Second cmd args

$3 ===> Third cmd args

$* ===> All Cmd Args

========== cmdargs.sh ==========

```
#! /bin/bash

echo $#

echo $0

echo $1

echo $2

echo $*
```

==== Execution : sh cmdargs.sh ashok it ==========

```
#!/bin/bash

#to access the arguments

echo "First argument is $1"
echo "Second argument is $2"
```

```
[paul@redhat01 myscripts]$ bash 26_args.sh
First argument is
Second argument is
```

এখানে যেহেতু আমরা কোন arguments pass করিনি তাই আমাদের এখানে কোন Output আসেনি command_line arguement থেকে

```
[paul@redhat01 myscripts]$ bash 26_args.sh Raju
First argument is Raju
Second argument is
[paul@redhat01 myscripts]$ bash 26_args.sh Raju Sham
First argument is Raju
Second argument is Sham
```

এখানে যেহেতু আমরা কোন arguments pass করেছি তাই আমাদের এখানে Output এ command line arguement থেকে add হয়ে গেছে।

MPRASHANT

ARGUMENTS IN SCRIPT...

```
for arg in $@
do
        echo "Argument is $arg"
done
```

```bash
#to access the arguments

echo "First argument is $1"
echo "Second argument is $2"

echo "All the arguments are - $@"
echo "Number of arguments are - $#"

#For loop to access the values from arguments

for filename in $@
do
        echo "Copying file - $filename"
done
```

```
[paul@redhat01 myscripts]$ bash 26_args.sh test.csv myfile.csv
First argument is test.csv
Second argument is myfile.csv
All the arguments are - test.csv myfile.csv
Number of arguments are - 2
Copying file - test.csv
Copying file - myfile.csv
```

MPRASHANT

SHIFTING ARGUMENTS

SHIFT

When we pass multiple arguments, we can shift.

```
A B C
shift
B C
```

```
#!/bin/bash

# To create a user, provide username and description

echo "Creating user"
echo "Username is $1"

echo "Description is $2"
```

```
[paul@redhat01 myscripts]$ vi 27_shift_args.sh
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ bash 27_shift_args.sh paul TEST
Creating user
Username is paul
Description is TEST
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ bash 27_shift_args.sh alex test user for QA team
Creating user
Username is alex
Description is test
[paul@redhat01 myscripts]$ bash 27_shift_args.sh alex "test user for QA team"
Creating user
Username is alex
Description is test user for QA team
```

```
#!/bin/bash

# To create a user, provide username and description

echo "Creating user"
echo "Username is $1"

shift
echo "Description is $@"
```

```
[paul@redhat01 myscripts]$ bash 27_shift_args.sh alex test user for QA team
Creating user
Username is alex
Description is test user for QA team
```

যদি আমরা কোন multiple string কে আমরা pass করতে চাই সেক্ষেত্রে আমাদের "" or shift use করতে হবে

Other Useful Concepts

MPRASHANT

USEFUL CONCEPTS...

break - to stop the loop

continue - to stop current iteration of loop and start next iteration

```
no=6

for i in 1 2 3 4 5 6 7 8 9
do
        #Break the loop if no. found
        if [[ $no -eq $i ]]
        then
                echo "$no is found!!!!"
                break
        fi
        echo "Number is $i"
done
```

```
Number is 1
Number is 2
Number is 3
Number is 4
Number is 5
6 is found!!!!
```

```
#!/bin/bash

#example of using continue in loop
# Suppose we only need to print odd no.

for i in 1 2 3 4 5 6 7 8 9 10
do
        let r=$i%2
        if [[ $r -eq 0 ]]
        then
                continue
        fi
        echo "odd no. is $i"
done
```

```
odd no. is 1
odd no. is 3
odd no. is 5
odd no. is 7
odd no. is 9
```

MPRASHANT

USEFUL CONCEPTS...

sleep - to create delay between two executions ex: sleep 1s/1m

exit - to stop script at a point

exit status $? - gives you status of previous command if that was successful

```bash
#!/bin/bash

#to access the arguments

if [[ $# -eq 0 ]]
then
        echo "Please provide atleast one argument"
        exit 1
fi

echo "First argument is $1"
echo "Second argument is $2"

echo "All the arguments are - $@"
echo "Number of arguments are - $#"

#For loop to access the values from arguments

for filename in $@
do
```

```
[paul@redhat01 myscripts]$ bash 26_args.sh
Please provide atleast one argument
```

```
[paul@redhat01 myscripts]$ bash 26_args.sh Paul
First argument is Paul
Second argument is
All the arguments are - Paul
Number of arguments are - 1
Copying file - Paul
```

```bash
#!/bin/bash

read -p "which site you want to check? " site

ping -c 1 $site
#sleep 1s

if [[ $? -eq 0 ]]
then
        echo "Successfully connected to $site"
else
        echo "Unable to connect $site"
fi
```

```
[paul@redhat01 myscripts]$ ping -c 1 www.google.com
PING www.google.com (142.251.39.4) 56(84) bytes of data.
64 bytes from bud02s37-in-f4.1e100.net (142.251.39.4): icmp_se
e=14.4 ms

--- www.google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 14.362/14.362/14.362/0.000 ms
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ echo $?
0
[paul@redhat01 myscripts]$ ping -c 1 localhost.com
PING localhost.com (74.125.224.72) 56(84) bytes of data.

--- localhost.com ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

[paul@redhat01 myscripts]$ echo $?
1
```

```bash
#!/bin/bash

read -p "which site you want to check? " site

ping -c 1 $site
sleep 5s

if [[ $? -eq 0 ]]
then
        echo "Successfully connected to $site"
else
        echo "Unable to connect $site"
fi
```

```
[paul@redhat01 myscripts]$ bash 30_connectivity_check.sh
which site you want to check? www.google.com
PING www.google.com (142.251.39.36) 56(84) bytes of data.
64 bytes from bud02s38-in-f4.1e100.net (142.251.39.36): icmp
me=16.7 ms

--- www.google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 16.740/16.740/16.740/0.000 ms
Successfully connected to www.google.com
```

```bash
#!/bin/bash

read -p "which site you want to check? " site

ping -c 1 $site
#sleep 1s

if [[ $? -eq 0 ]]
then
        echo "Successfully connected to $site"
else
        echo "Unable to connect $site"
fi
```

```
[paul@redhat01 myscripts]$ vi 30_connectivity_check.sh
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ bash 30_connectivity_check.sh
which site you want to check? localhost.com
PING localhost.com (74.125.224.72) 56(84) bytes of data.

--- localhost.com ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms

Unable to connect localhost.com
```

MPRASHANT

USEFUL CONCEPTS...

basename - strip directory info and only give filename

```
[paul@redhat01 myscripts]$ basename /home/paul/myscripts/test.csv
test.csv
```

dirname - strip the filename and gives directory path

```
[paul@redhat01 myscripts]$ dirname /home/paul/myscripts/test.csv
/home/paul/myscripts
```

realpath - gives you full path for a file

```
[paul@redhat01 myscripts]$ realpath test.csv
/home/paul/myscripts/test.csv
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ realpath hi
/home/paul/myscripts/hi
```

MPRASHANT

CHECK IF FILE/DIR EXIST

```
if [ -d folder_name] If folder exists

 [ ! -d folder_name] If folder not exists
```

```
if [ -f file_name] If file exists

if [ ! -f file_name] If file not exists
```

```
#!/bin/bash

FILEPATH="/home/pauly/myscripts/test.csv"

if [[ -f $FILEPATH ]]
then
        echo "File exist"
else
        echo "File not exist"
        exit 1
fi
```

```
[paul@redhat01 myscripts]$ vi 31_file_exit_check.sh
[paul@redhat01 myscripts]$ bash 31_file_exit_check.sh
File exist
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ vi 31_file_exit_check.sh
[paul@redhat01 myscripts]$ bash 31_file_exit_check.sh
File not exist
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ echo $?
1
```

```
#!/bin/bash

FILEPATH="/home/paul/myscripts/prashant.test"

if [[ -f $FILEPATH ]]
then
        echo "File exist"
else
        echo "Creating file now"
        touch $FILEPATH
fi
```

```
[paul@redhat01 myscripts]$ bash 31_file_exit_check.sh
Creating file now
[paul@redhat01 myscripts]$ bash 31_file_exit_check.sh
File exist
```

```
-rw-rw-r--. 1 paul paul   0 Jul 27 01:46 prashant.test
```

MPRASHANT

BASH VARIABLES...

RANDOM - A random integer between 0 and 32767 is generated

UID - User ID of the user logged in

```
[paul@redhat01 myscripts]$ echo $RANDOM
15996
[paul@redhat01 myscripts]$ echo $RANDOM
26385
[paul@redhat01 myscripts]$ echo $RANDOM
9800
[paul@redhat01 myscripts]$ echo $RANDOM
26377
[paul@redhat01 myscripts]$ echo $RANDOM
28556
[paul@redhat01 myscripts]$ echo $RANDOM
21655
[paul@redhat01 myscripts]$ echo $RANDOM
26786
[paul@redhat01 myscripts]$ echo $RANDOM
32484
```

```
[paul@redhat01 myscripts]$ echo $UID
1000
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ su -
Password:
Last login: Mon Jul 24 08:26:26 EEST 2023 from 10.211.55.2 on pts/1
[root@redhat01 ~]#
[root@redhat01 ~]# echo $UID
0
```

```bash
#!/bin/bash

#Generating a random no. between 1 to 6

NO=$(( $RANDOM%6 + 1 ))
echo "Number is $NO"
```

```
[paul@redhat01 myscripts]$ bash 32_dice.sh
Number is 5
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ bash 32_dice.sh
Number is 3
[paul@redhat01 myscripts]$ bash 32_dice.sh
Number is 2
[paul@redhat01 myscripts]$ bash 32_dice.sh
Number is 2
[paul@redhat01 myscripts]$ bash 32_dice.sh
Number is 4
[paul@redhat01 myscripts]$ bash 32_dice.sh
Number is 3
[paul@redhat01 myscripts]$ bash 32_dice.sh
Number is 2
[paul@redhat01 myscripts]$ bash 32_dice.sh
Number is 5
[paul@redhat01 myscripts]$ bash 32_dice.sh
Number is 1
```

```
#!/bin/bash


#checking if the user is root or not

if [[ $UID -eq 0 ]]
then
        echo "User is root"
else
        echo "User is not root"
fi
```

```
[paul@redhat01 myscripts]$ vi 33_root_usercheck.sh
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ bash 33_root_usercheck.sh
User is not root
[paul@redhat01 myscripts]$ su -
Password:
Last login: Thu Jul 27 01:49:15 EEST 2023 on pts/0
[root@redhat01 ~]#
[root@redhat01 ~]# bash /home/paul/myscripts/33_root_usercheck.sh
User is root
```

Redirection in scripts

> >>

```
[paul@redhat01 myscripts]$ date > all_files.txt
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ cat all_files.txt
Thursday 27 July 2023 02:25:01 AM EEST
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ pwd >> all_files.txt
[paul@redhat01 myscripts]$ hostname >> all_files.txt
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ cat all_files.txt
Thursday 27 July 2023 02:25:01 AM EEST
/home/paul/myscripts
redhat01
```

MPRASHANT

WHAT IS /DEV/NULL

In case if you don’t wanna print the output of a command on terminal or write in a file,

We can redirect the output to /dev/null

Example:

```
#cd /root &> /dev/null
```

```
[paul@redhat01 myscripts]$ cd /root/
bash: cd: /root/: Permission denied
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ cd /root/ &> /dev/null
```

```
#!/bin/bash

read -p "which site you want to check? " site

ping -c 1 $site &> /dev/null
#sleep 1s

if [[ $? -eq 0 ]]
then
        echo "Successfully connected to $site"
else
        echo "Unable to connect $site"
fi
```

```
[paul@redhat01 myscripts]$ bash 30_connectivity_check.sh
which site you want to check? www.google.com
Successfully connected to www.google.com
```

MPRASHANT

PRINT NAME OF THE SCRIPT

```
echo "The name of the script is: ${0}"
```

```
#!/bin/bash


echo "Name of this script is ${0}"
```

```
[paul@redhat01 myscripts]$ vi 35_scriptname.sh
[paul@redhat01 myscripts]$ bash 35_scriptname.sh
Name of this script is 35_scriptname.sh
```

MPRASHANT

LOG MESSAGES..

If you want to maintain the logging for your script, you can use logger in your script.

You can find the logs under /var/logs/messages

Example: #logger “Hey Buddy”

```
#!/bin/bash

#example of logging

logger "This is log from ${0}"
```

```
Jul 27 02:36:32 redhat01 paul[135777]: This is log from 36_logger.sh
```

MPRASHANT

DEBUGGING SCRIPTS

If We can enable the debugging of the script using below in the script

```
set -x
```

```
#!/bin/bash

set -x
#to access the arguments

if [[ $# -eq 0 ]]
then
        echo "Please provide atleast one argument"
        exit 1
fi

echo "First argument is $1"
echo "Second argument is $2"

echo "All the arguments are - $@"
echo "Number of arguments are - $#"

#For loop to access the values from arguments

for filename in $@
```

```
[paul@redhat01 myscripts]$ vi 26_args.sh
[paul@redhat01 myscripts]$ bash 26_args.sh
+ [[ 0 -eq 0 ]]
+ echo 'Please provide atleast one argument'
Please provide atleast one argument
+ exit 1
[paul@redhat01 myscripts]$
[paul@redhat01 myscripts]$ bash 26_args.sh Paul
+ [[ 1 -eq 0 ]]
+ echo 'First argument is Paul'
First argument is Paul
+ echo 'Second argument is '
Second argument is
+ echo 'All the arguments are - Paul'
All the arguments are - Paul
+ echo 'Number of arguments are - 1'
Number of arguments are - 1
+ for filename in $@
+ echo 'Copying file - Paul'
Copying file - Paul
```

MPRASHANT

DEBUGGING SCRIPTS

If We want to exit our script when a command fail

```
set -e
```

```
#!/bin/bash

pwd
date
cd /root
hostname
```

```
[paul@redhat01 myscripts]$ bash 37_debugging.sh
/home/paul/myscripts
Thursday 27 July 2023 02:43:01 AM EEST
37_debugging.sh: line 5: cd: /root: Permission denied
redhat01
```

```
#!/bin/bash

set -e

pwd
date
cd /root
hostname
```

```
[paul@redhat01 myscripts]$ vi 37_debugging.sh
[paul@redhat01 myscripts]$ bash 37_debugging.sh
/home/paul/myscripts
Thursday 27 July 2023 02:43:39 AM EEST
37_debugging.sh: line 7: cd: /root: Permission denied
```

Running Script in background

```
nohup ./script.sh &
```

Automate our Script

At or Crontab

For scheduling only one time, use AT

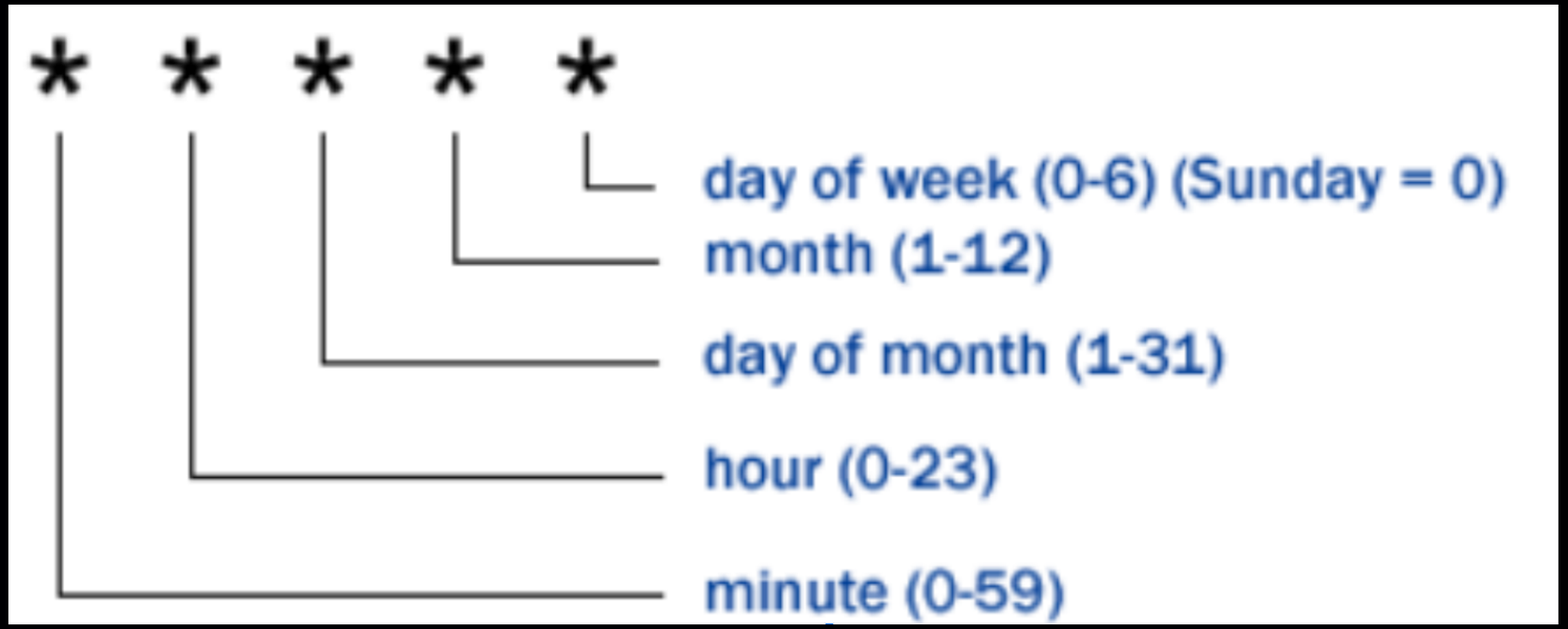
```
at 12:09 PM
  <your_command>
Ctrl + D
```

`atq` to check scheduled job
`atrm <id>` to remove the schedule

To check the existing jobs - crontab -l

To add new job - crontab -e

```
* * * * cd /home/paul/scripts && ./create_file.sh
```


* * * * *
day of week (0-6) (Sunday = 0)
month (1-12)
day of month (1-31)
hour (0-23)
minute (0-59)

Projects

MPRASHANT

1 Monitoring free RAM space

MPRASHANT

2

Monitoring free DISK space and sent an Alert Email

MPRASHANT

3

Access Remote server and perform some actions

MPRASHANT