# \*\*Bash Shell Scripting\*\*

![](https://lh7-us.googleusercontent.com/7ibztAUH9YGUTbFhsy2gfL0QRW75HHbUCIMa-s\_-c5ganCn1i6Xoh6gv5d2GjtDrnxw-\_9vf91nk3J8pW2At9MQERlZm\_hJJWbzUwqPBJUlQDha3IsC\_EE1JxHEuWtUo0oIxWFgvtALWBvdL00hSTA)

![](https://lh7-us.googleusercontent.com/mVhKrK79bHLqPTGIjd3FND9OhxbtgPJRxMc7-BclQqFBhfyd4QBVYSVasTLG0Bw\_ilGsAEVqBo1xZMl-xbXn3B40tcJACkLw1RzT89bc9Wn4Io3hun2ux2P7-G4wJB8Vzb0Q5Yz\_HXeVXLlA6x7Gfg)

# \*\*By :- Himanshu (December Intern)\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*Table of Content\*\*

[1. Introduction ](#introduction)

[Definition of Bash scripting ](#definition-of-bash-scripting)

[2. Advantages of Bash scripting ](#advantages-of-bash-scripting)

[➔ Automation ](#automation)

[➔ Portability ](#portability)

[➔ Flexibility ](#flexibility)

[➔ Accessibility ](#accessibility)

[➔ Integration ](#integration)

[➔ Debugging ](#debugging)

[3. How to Get Started with Bash Scripting ](#how-to-get-started-with-bash-scripting)

[4. Basic Bash commands ](#basic-bash-commands)

##

# 1.  \*\*Introduction\*\*<a id="introduction"></a>

- Shell Scripting consists of a set of commands to perform a task.

- All commands execute sequentially.

- Tasks like file manipulation, program execution, user interaction, automation of tasks etc can be done by Shell Scripting.

## Definition of Bash scripting<a id="definition-of-bash-scripting"></a>

- BASH stands for \*\*Bourne Again SHell\*\*

- A bash script is a file containing a sequence of commands that are executed by the bash program line by line. It allows us to perform a series of actions, such as navigating to a specific directory, creating a folder, and launching a process using the command line.

\* By saving these commands in a script, we can repeat the same sequence of steps multiple times and execute them by running the script.

# 2.  \*\*Advantages of Bash scripting\*\*<a id="advantages-of-bash-scripting"></a>

## a) \*\*Automation\*\*<a id="automation"></a>

\* It allows us to automate repetitive task and processes, saving time and reducing the risk of errors that can occur with manual execution.

## b) \*\*Portability\*\*<a id="portability"></a>

\* Shell scripts can run on various platforms and operating systems, like UNIX, Linux, macOS, and even Windows through the use of emulators or virtual machines.

## c) \*\*Flexibility\*\*<a id="flexibility"></a>

\* Shell scripts are highly customizable and can be easily modified to suit specific requirements.

- They can also be combined with other programming languages or utilities to create more powerful scripts.

## d) \*\*Accessibility\*\*<a id="accessibility"></a>

- Shell scripts are easy to write and don't require any special tools or software.

\* They can be edited using any text editor like ( nano,vim,vi ), and most operating systems have a built-in shell interpreter.

## e) \*\*Integration\*\*<a id="integration"></a>

\* Shell scripts can be integrated with other tools and applications, such as databases, web servers, and cloud services, allowing for more complex automation and system management tasks.

## f) \*\*Debugging\*\*<a id="debugging"></a>

\* Shell scripts are easy to debug, and most shells have built-in debugging and error-reporting tools that can help identify and fix issues quickly.

# 3.  \*\*How to Get Started with Bash Scripting\*\*<a id="how-to-get-started-with-bash-scripting"></a>

To start a Bash script in Linux, we need to follow these general steps:

1. Create a Bash script: Use a text editor, such as `nano`, `vim`, or `gedit`, to create a new file and write your Bash script. For example:

```

nano myscript.sh

```

Inside the editor, write your script. Here's a simple example:

```

#!/bin/bash

echo "Hello, World!"

```

Save the file and exit the text editor.

2. Make the script executable: You need to make the script executable to run it. Use the `chmod` command to do this:

```

chmod +x myscript.sh

```

3. Run the script: We can execute the script by typing its name preceded by `bash` in the terminal:

```

bash myscript.sh

```

This assumes that the script is in the current directory. If it's in a different directory, we need to provide the full or relative path.

Keep in mind that the first line (`#!/bin/bash`) is called a \*\*SHEBANG\*\*, and it specifies the interpreter (in this case, Bash) to be used to run the script. It's necessary for the script to be interpreted correctly.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# :- \*\*Here are some Bash Shell Script code :-\*\*

i) \*\*Basic code\*\*

```

nano basic.sh

```

```

#!/bin/bash

echo "what is the number ?"

read k

echo "The number is :-" $k

```

\*\*OUTPUT\*\*

```

bash basic.sh

```

\*\*\*

what is the number ?

23

The number is :- 23

\*\*\*

\*\*NOTE ➖\*\*

- `echo` command is used to print the statement.

- `read` command is used to enter the value.

- `$` means to find a specific value that we enter with the help of the read command.

- To run the program we have to write “\*\*bash filename.sh\*\*”, here `.sh` is the extension to run the program.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ii) \*\*Conditional Statement\*\*

```

nano condition.sh

```

```

#!/bin/bash

echo "Enter the number"

read num

if [ $num -lt 1000 ];

then

echo "Number is correct"

else

echo "Number is incorrect"

fi

```

\*\*OUTPUT\*\*

```

bash condition.sh

```

\*\*\*

Enter the number

12345

\*\*Number is incorrect\*\*

```

bash condition.sh

```

Enter the number

123

\*\*Number is correct\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

iii) \*\*OR (`||`) Operator\*\*

```

nano or.sh

```

```

#!/bin/bash

echo "Now enter the details of user"

echo "Enter the username"

read user

echo "Enter the password"

read pass

if [[ ( $user == "Kartikay" || $pass == "301224") ]];

then

echo "user is valid"

else

echo "user is invalid"

fi

```

\*\*OUTPUT\*\*

```

bash or.sh

```

\*\*\*\*

Now enter the details of user

Enter the username

Kartik

Enter the password

301224

\*\*user is valid\*\*

```

bash or.sh

```

Now enter the details of user

Enter the username

kartikay

Enter the password

301224

\*\*user is valid\*\*

\*\*\*\*

\*\*\*\*

\*\*\*\*

iv) \*\*AND (`&`) Operator\*\*

```

nano and.sh

```

```

#!/bin/bash

echo "Now enter the details of user"

echo "Enter the username"

read user

echo "Enter the password"

read pass

if [[ ( $user == "Kartikay" && $pass == "301224") ]];

then

echo "user is valid"

else

echo "user is invalid"

fi

```

\*\*OUTPUT\*\*

```

bash and.sh

```

\*\*\*

Now enter the details of user

Enter the username

kartikay

Enter the password

301224

\*\*user is invalid\*\*

```

bash and.sh

```

Now enter the details of user

Enter the username

Kartikay

Enter the password

301224

\*\*user is valid\*\*

\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

v) \*\*CASE Option\*\*

```

nano case.sh

```

```

#!/bin/bash

echo "Enter the assigned Single digit no. to check Employee details"

read val

case $val in

1)

echo "Name – Ram  Emp ID – 101 ";;

2)

echo "Name – Ramesh  Emp ID – 103 ";;

3)

echo "Name – Shyam  Emp ID – 102 ";;

4)

echo "Name – Vishal Emp ID – 104 ";;

5)

echo "Name – Aman  Emp ID – 105 ";;

\*)

echo "Sorry! Invalid choice";;

esac

```

\*\*OUTPUT\*\*

\*\*\*\*\*\*\*\*\*\*

```

bash case.sh

```

Enter the assigned Single digit no. to check Employee details

2

\*\*Name – Ramesh  Emp ID – 103\*\*

```

bash case.sh

```

Enter the assigned Single digit no. to check Employee details

\*\*\\*\*\*

\*\*Sorry! Invalid choice\*\*

\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*

vi) \*\*Looping\*\*

This is a process in which all the statements will run till condition is true

\*\*(a) For Loop\*\*

```

nano loop.sh

```

```

#!/bin/bash

for (( count==0; count<10; count++; ))

do

echo $count

done

```

\*\*OUTPUT\*\*

\*\*\*\*\*\*\*\*\*\*

```

bash loop.sh

```

1

2

3

4

5

6

7

8

9

\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*

\*\*(b) While Loop\*\*

```

nano while.sh

```

```

#!/bin/bash

counter=1

while [ $counter -le 7 ]

do

    echo $counter

    ((counter++))

done

```

\*\*OUTPUT\*\*

\*\*\*\*\*

```

bash while.sh

```

1

2

3

4

5

6

7

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

# \*\*To search file with the help of scripting\*\*

```

nano search.sh

```

```

#!/bin/bash

echo "Enter the file name you want to search"

read name

if [ -f $name ];

then

echo "File is exists"

else

echo "File does not exists"

fi

```

\*\*Note ➖\*\*

- Here `f` helps to find the file from our system

\*\*OUTPUT\*\*

```

bash search.sh

```

Enter the file name you want to search

kaushik

\*\*File is exists\*\*

```

bash search.sh

```

Enter the file name you want to search

kartik

\*\*File does not exists\*\*

\*\*Note ➖\*\*

- Remember that file must already exist in your system.

\*\*\*\*

\*\*\*\*

\*\*\*\*

# 4.  \*\*Basic Bash commands\*\*<a id="basic-bash-commands"></a>

1)      cd: Change the directory to a different location.

2)      ls: List the contents of the current directory.

3)      mkdir: Create a new directory.

4)      touch: Create a new empty file.

5)      rm: Remove a file or directory.

6)      cp: Copy a file or directory.

7)      mv: Move or rename a file or directory.

8)      echo: Print text to the terminal.

9)      cat: Concatenate and print the contents of a file.

10) read: This command helps us to print the value.