

ASSESSMENT-2

Title: Bash Shell Basics

NAME: PAWAR ADWYAIT SHIVAJI

REG. NO.: 20BCE2088

COURSE: CYBERSECURITY AND ETHICAL HACKING

SMARTBRIDGE EXTERNSHIP PROGRAM

VIT Email: pawaradwyait.shivaji2020@vitstudent.ac.in

DATE: 26 MAY,2023

Task 1: File and Directory Manipulation

1. Create a directory called "my_directory".

```
(adwyait@kali)-[~]  
$ mkdir my_directory
```

2. Navigate into the "my_directory".

```
(adwyait@kali)-[~]  
$ cd my_directory
```

3. Create an empty file called "my_file.txt".

```
(adwyait@kali)-[~/my_directory]  
$ touch my_file.txt
```

4. List all the files and directories in the current directory.

```
(adwyait@kali)-[~/my_directory]  
$ ls  
my_file.txt
```

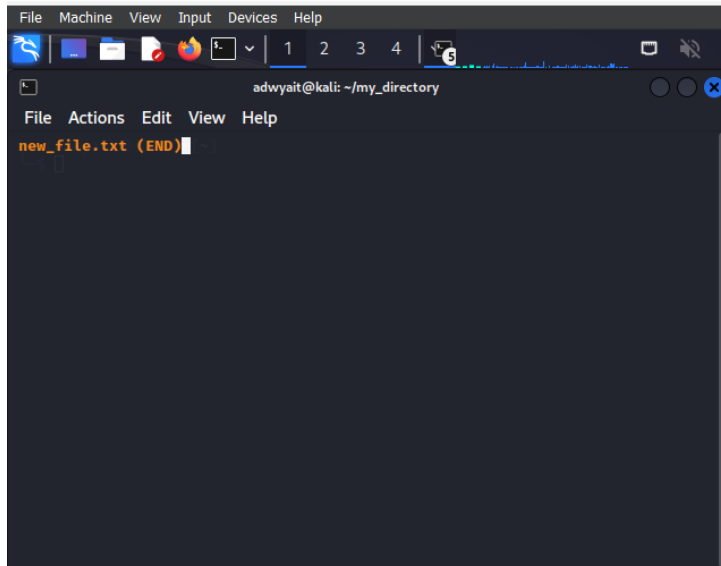
5. Rename "my_file.txt" to "new_file.txt".

```
(adwyait@kali)-[~/my_directory]  
$ mv my_file.txt new_file.txt
```

6. Display the content of "new_file.txt" using a pager tool of your choice.

```
(adwyait@kali)-[~/my_directory]  
$ less new_file.txt
```

Using less cmd to display the file content



7. Append the text "Hello, World!" to "new_file.txt".

```
(adwyait@kali)-[~/my_directory]
$ echo "Hello,World!" >> new_file.txt
dquote> "
Hello,World >> new_file.txt
```

8. Create a new directory called "backup" within "my_directory".

```
(adwyait@kali)-[~/my_directory]
$ mkdir backup
```

9. Move "new_file.txt" to the "backup" directory.

```
(adwyait@kali)-[~/my_directory]
$ mv new_file.txt backup/
```

10. Verify that "new_file.txt" is now located in the "backup" directory.

```
(adwyait@kali)-[~/my_directory]
$ ls backup/
new_file.txt
```

11. Delete the "backup" directory and all its contents.

```
(adwyait@kali)-[~/my_directory]
$ rm -r backup/
```

Task 2: Permissions and Scripting

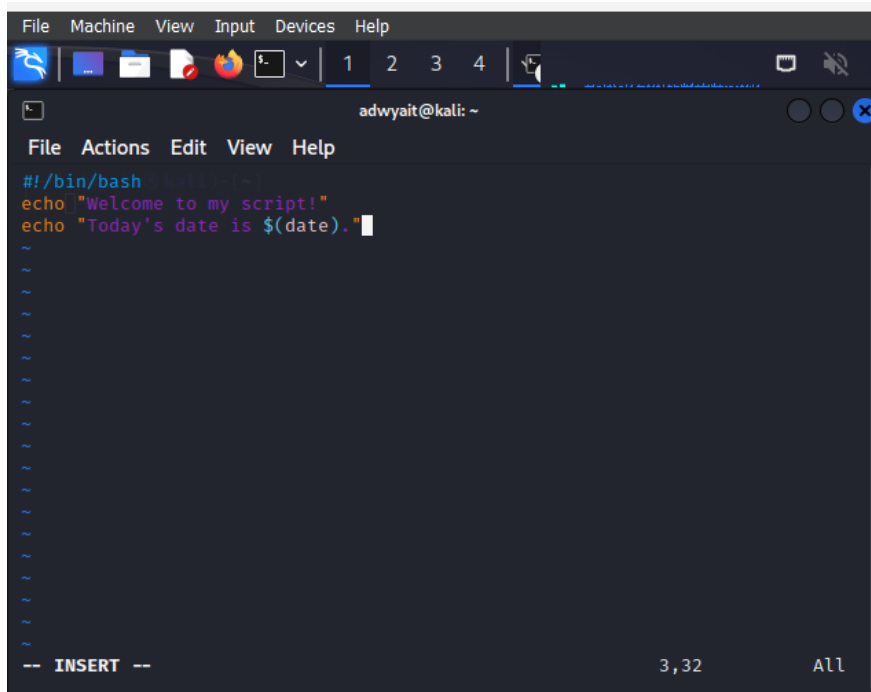
1. Create a new file called "my_script.sh".

```
(adwyait@kali)-[~]  
$ touch my_script.sh
```

2. Edit "my_script.sh" using a text editor of your choice and add the following lines:

- a. bash
- b. #!/bin/bash
- c. echo "Welcome to my script!"
- d. echo "Today's date is \$(date)." Save and exit the file.

```
(adwyait@kali)-[~]  
$ vim my_script.sh
```



The screenshot shows a terminal window with a dark background. At the top, there's a menu bar with 'File', 'Machine', 'View', 'Input', 'Devices', and 'Help'. Below the menu bar, there's a toolbar with various icons. The main area of the terminal shows the vim editor interface. The prompt is 'adwyait@kali: ~'. The editor has a menu bar with 'File', 'Actions', 'Edit', 'View', and 'Help'. The content of the file 'my_script.sh' is displayed in a monospaced font:
#!/bin/bash
echo "Welcome to my script!"
echo "Today's date is \$(date)."
The status bar at the bottom shows '-- INSERT --' on the left, '3,32' in the center, and 'All' on the right.

3. Make "my_script.sh" executable.

```
(adwyait@kali)-[~]  
$ chmod +x my_script.sh
```

4. Run "my_script.sh" and verify that the output matches the expected result.

```
(adwyait@kali)-[~]  
$ ./my_script.sh  
Welcome to my script!  
Today's date is Friday 26 May 2023 10:53:43 PM IST.
```

The given instructions create a Bash script file, "my_script.sh", which prints a welcome message and the current date when executed. The script file is made executable using the **chmod** command, and then it is run to verify the output.

Task 3: Command Execution and Pipelines

1. List all the processes running on your system using the "ps" command.

```
(adwyait@kali)-[~]  
$ ps  
  PID TTY          TIME CMD  
 3229 pts/3        00:00:06 zsh  
 15166 pts/3        00:00:00 ps
```

A list of processes associated with the current terminal session.

2. Use the "grep" command to filter the processes list and display only the processes with "bash" in their name.

```
(adwyait@kali)-[~]  
$ ps aux | grep bash  
adwyait 15702 0.0 0.2 6332 2072 pts/3 S+ 22:57 0:00 grep --co  
lor=auto bash
```

This command will display the processes list, and **grep** will filter out the lines that contain the word "bash".

3. Use the "wc" command to count the number of lines in the filtered output.

```
(adwyait@kali)-[~]  
$ ps aux | grep bash | wc -l  
1
```

The **wc -l** command counts the number of lines in the output from the previous **grep** command. The result will be the number of processes that have "bash" in their name.