

NAME: KAMALESH D
REG. NO.: 20MIS0342
CAMPUS: VIT – VELLORE
PROGRAM: M. TECH INT. SOFTWARE ENGINEERING
COURSE TAKEN: CYBER SECURITY & ETHICAL HACKING

Assignment – 2: Bash Shell Basics

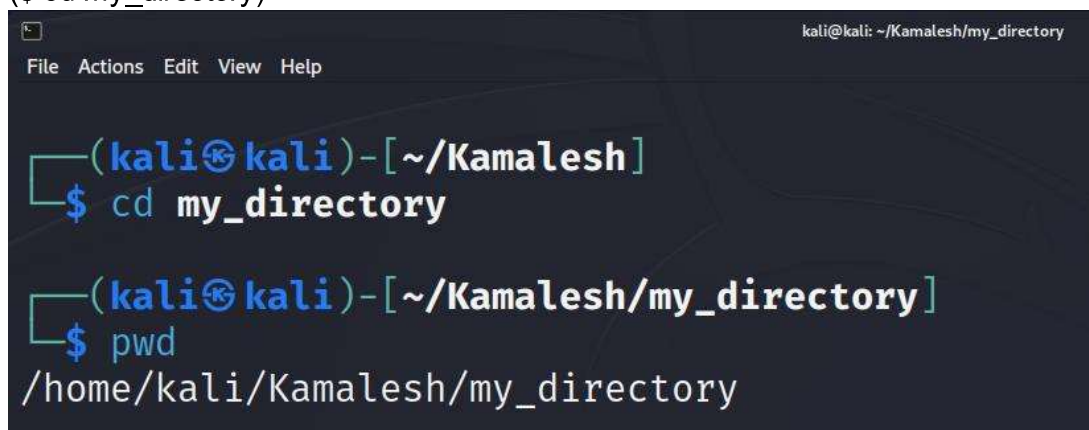
Task 1: File and Directory Manipulation

1. Create a directory called "my_directory".
(\$ mkdir my_directory)

A terminal window with a dark background and light blue text. The window has a menu bar with 'File', 'Actions', 'Edit', 'View', and 'Help'. The prompt is '(kali㉿kali)-[~/Kamalesh]'. The user enters '\$ mkdir my_directory'. The prompt changes to '(kali㉿kali)-[~/Kamalesh]'. The user enters '\$ ls'. The output 'my_directory' is displayed in light blue.

```
(kali㉿kali)-[~/Kamalesh]  
$ mkdir my_directory  
  
(kali㉿kali)-[~/Kamalesh]  
$ ls  
my_directory
```

2. Navigate into the "my_directory".
(\$ cd my_directory)

A terminal window with a dark background and light blue text. The window has a menu bar with 'File', 'Actions', 'Edit', 'View', and 'Help'. The title bar on the right says 'kali@kali: ~/Kamalesh/my_directory'. The prompt is '(kali㉿kali)-[~/Kamalesh]'. The user enters '\$ cd my_directory'. The prompt changes to '(kali㉿kali)-[~/Kamalesh/my_directory]'. The user enters '\$ pwd'. The output '/home/kali/Kamalesh/my_directory' is displayed in light blue.

```
(kali㉿kali)-[~/Kamalesh]  
$ cd my_directory  
  
(kali㉿kali)-[~/Kamalesh/my_directory]  
$ pwd  
/home/kali/Kamalesh/my_directory
```

3. Create an empty file called "my_file.txt".
(\$ touch my_file)

A terminal window with a dark background and light blue text. The title bar shows 'kali@kali: ~/Kamalesh/my_directory'. The menu bar includes 'File', 'Actions', 'Edit', 'View', and 'Help'. The prompt is '(kali@kali)-[~/Kamalesh/my_directory]'. The user enters '\$ touch my_file.txt'. The prompt repeats, and the user enters '\$ ls'. The output 'my_file.txt' is displayed.

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

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

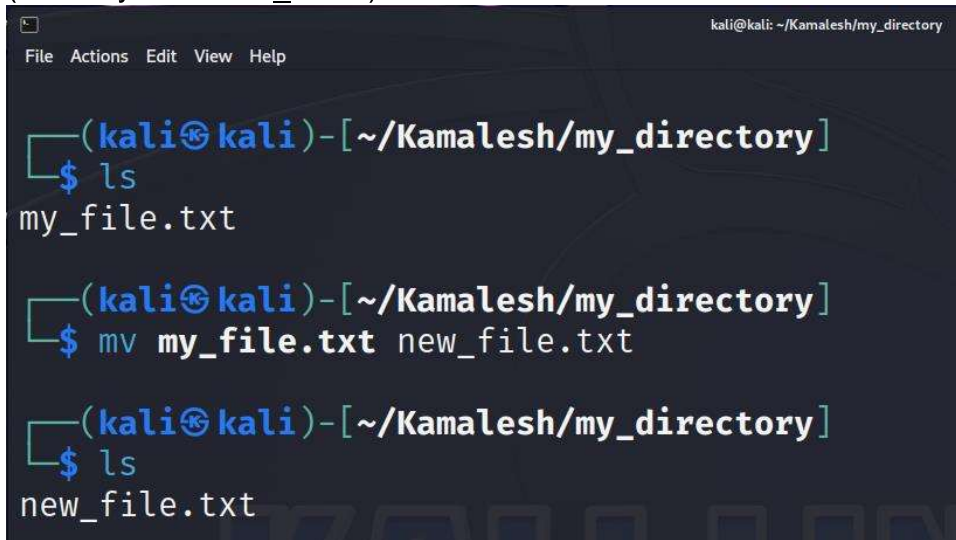
4. List all the files and directories in the current directory.
(\$ ls)

A terminal window with a dark background and light blue text. The title bar shows 'kali@kali: ~/Kamalesh/my_directory'. The menu bar includes 'File', 'Actions', 'Edit', 'View', and 'Help'. The prompt is '(kali@kali)-[~/Kamalesh/my_directory]'. The user enters '\$ pwd'. The output '/home/kali/Kamalesh/my_directory' is displayed. The prompt repeats, and the user enters '\$ ls'. The output 'my_file.txt' is displayed.

```
(kali@kali)-[~/Kamalesh/my_directory]
$ pwd
/home/kali/Kamalesh/my_directory

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

5. Rename "my_file.txt" to "new_file.txt".
(\$ mv my_file.txt new_file.txt)

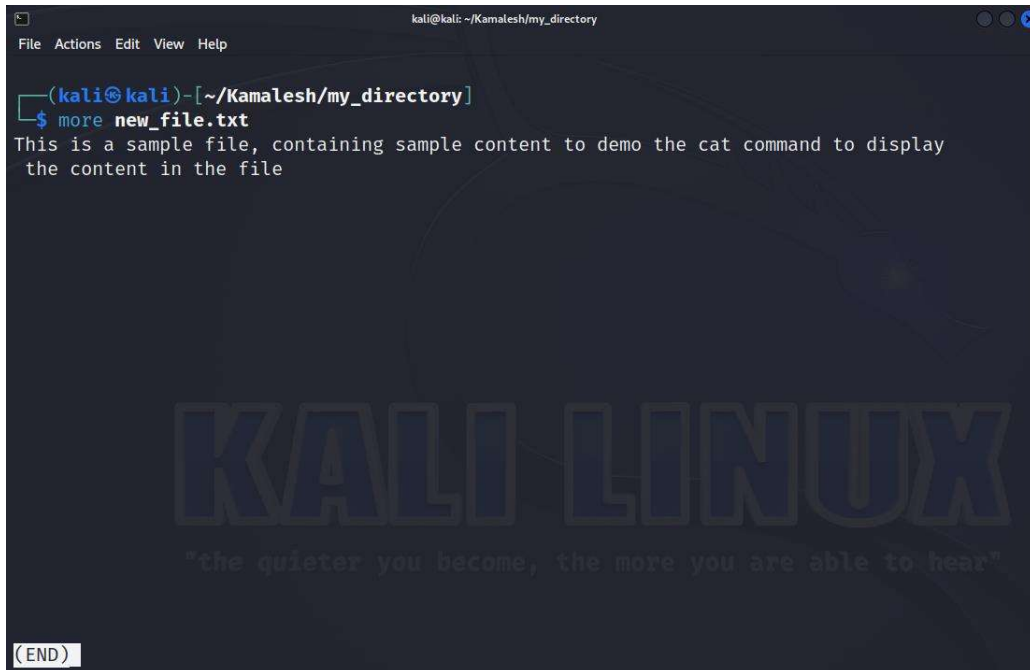
A terminal window with a dark background and light blue text. The title bar shows 'kali@kali: ~/Kamalesh/my_directory'. The menu bar includes 'File', 'Actions', 'Edit', 'View', and 'Help'. The prompt is '(kali@kali)-[~/Kamalesh/my_directory]'. The user enters '\$ ls'. The output 'my_file.txt' is displayed. The prompt repeats, and the user enters '\$ mv my_file.txt new_file.txt'. The prompt repeats again, and the user enters '\$ ls'. The output 'new_file.txt' is displayed.

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

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

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

6. Display the content of "new_file.txt" using a pager tool of your choice.
(used '`$ cat > new_file.txt`' to write contents in the text file;
used pager command '`$ more`' to display the content within pager format)

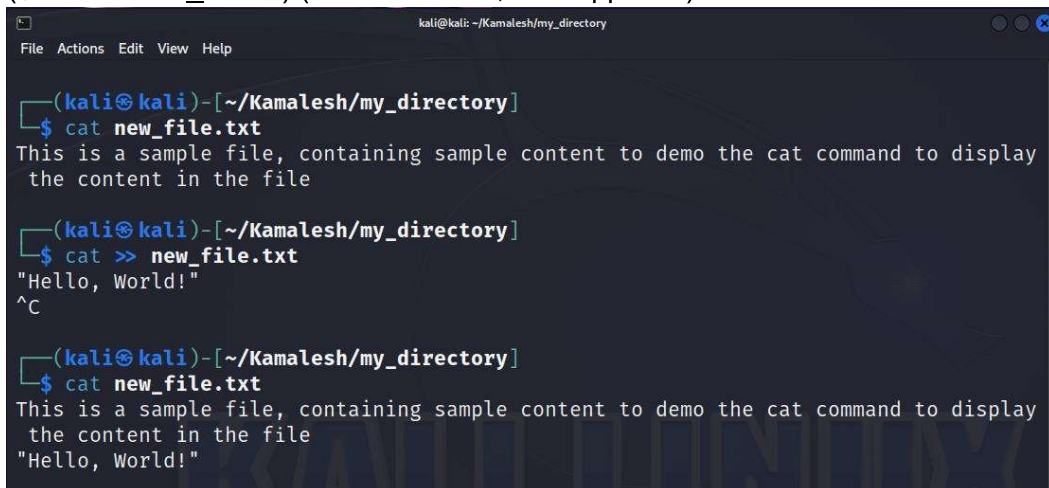


```
kali@kali: ~/Kamalesh/my_directory
File Actions Edit View Help

(kali@kali)-[~/Kamalesh/my_directory]
$ more new_file.txt
This is a sample file, containing sample content to demo the cat command to display
the content in the file

(END)
```

7. Append the text "Hello, World!" to "new_file.txt".
(\$ cat >> new_file.txt) (> - overwrites ; >> - appends)



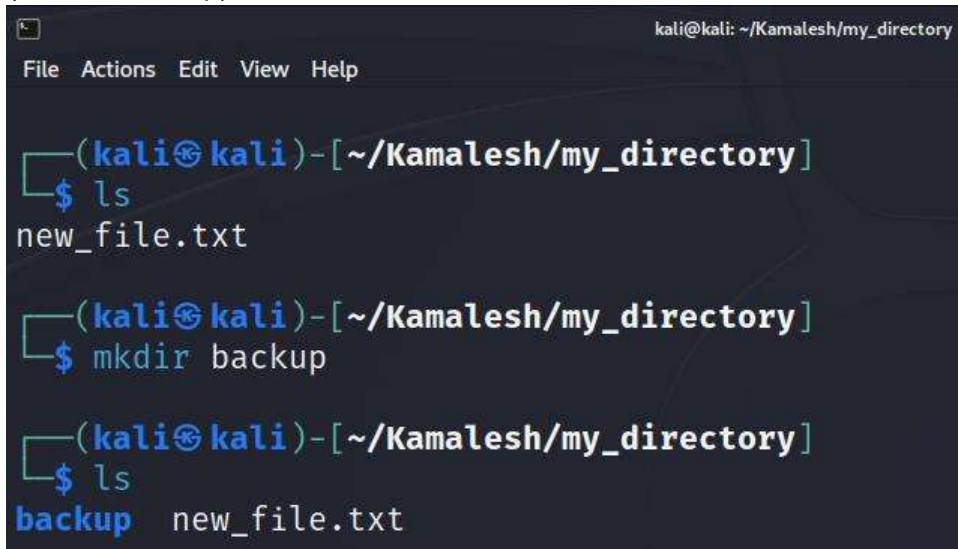
```
kali@kali: ~/Kamalesh/my_directory
File Actions Edit View Help

(kali@kali)-[~/Kamalesh/my_directory]
$ cat new_file.txt
This is a sample file, containing sample content to demo the cat command to display
the content in the file

(kali@kali)-[~/Kamalesh/my_directory]
$ cat >> new_file.txt
"Hello, World!"
^C

(kali@kali)-[~/Kamalesh/my_directory]
$ cat new_file.txt
This is a sample file, containing sample content to demo the cat command to display
the content in the file
"Hello, World!"
```

8. Create a new directory called "backup" within "my_directory".
(\$ mkdir backup)



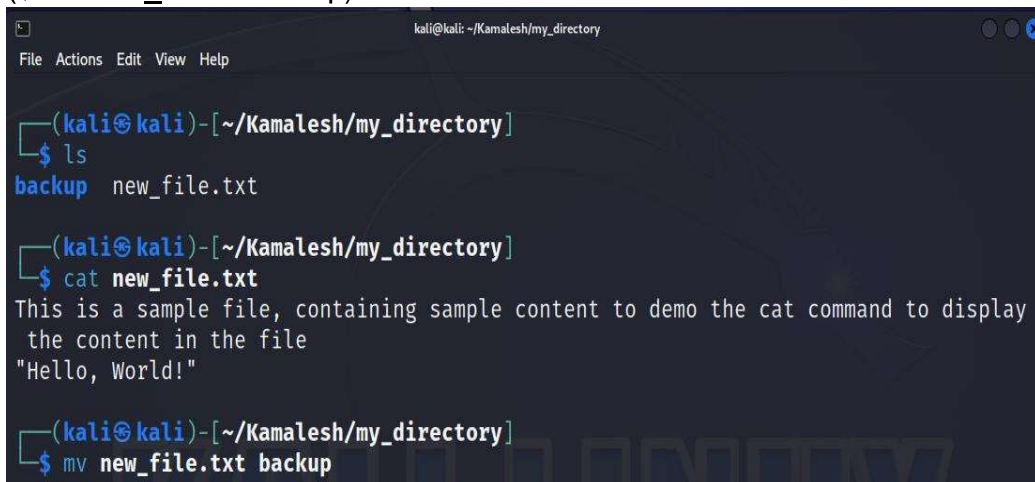
```
kali@kali: ~/Kamalesh/my_directory
File Actions Edit View Help

(kali@kali)-[~/Kamalesh/my_directory]
$ ls
new_file.txt

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

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

9. Move "new_file.txt" to the "backup" directory.
(\$ mv new_file.txt backup)



```
kali@kali: ~/Kamalesh/my_directory
File Actions Edit View Help

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

(kali@kali)-[~/Kamalesh/my_directory]
$ cat new_file.txt
This is a sample file, containing sample content to demo the cat command to display
the content in the file
"Hello, World!"

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

10. Verify that "new_file.txt" is now located in the "backup" directory.
(\$ cd backup – to navigate to the directory)

(\$ cat new_file.txt - to display the content of the file)

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

(kali@kali)-[~/Kamalesh/my_directory]
$ cd backup && ls
new_file.txt

(kali@kali)-[~/Kamalesh/my_directory/backup]
$ cat new_file.txt
This is a sample file, containing sample content to demo the cat command to display
the content in the file
"Hello, World!"
```

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

(\$ rm backup – wouldn't delete a non-empty directory)

(\$ rm -rf backup -The option -r deletes non-empty directories as well)

```
kali@kali: ~/Kamalesh/my_directory
File Actions Edit View Help

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

(kali@kali)-[~/Kamalesh/my_directory/backup]
$ cd .. && ls
backup

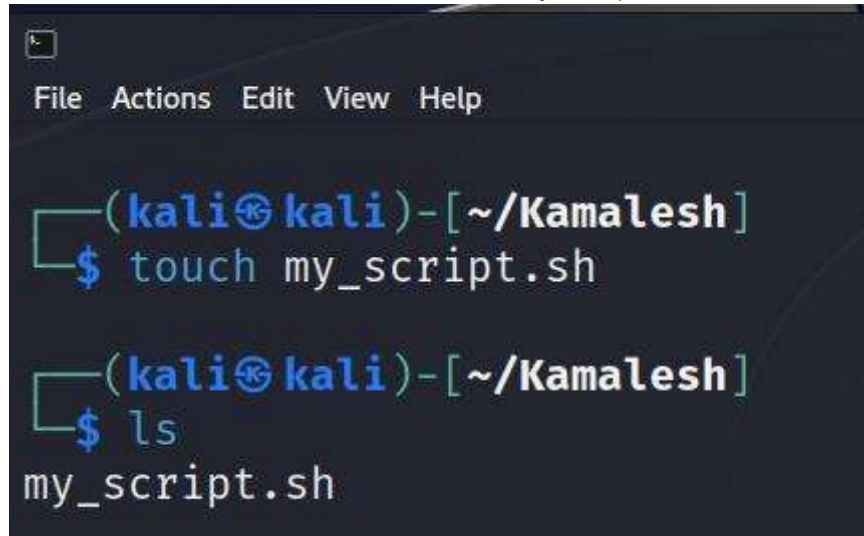
(kali@kali)-[~/Kamalesh/my_directory]
$ rm -rf backup

(kali@kali)-[~/Kamalesh/my_directory]
$ ls

(kali@kali)-[~/Kamalesh/my_directory]
$
```

Task 2: Permissions and Scripting

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

A terminal window with a dark background and light blue text. The window has a menu bar with 'File', 'Actions', 'Edit', 'View', and 'Help'. The prompt is '(kali㉿kali)-[~/Kamalesh]'. The user enters '\$ touch my_script.sh'. The prompt is '(kali㉿kali)-[~/Kamalesh]'. The user enters '\$ ls'. The output is 'my_script.sh'.

```
(kali㉿kali)-[~/Kamalesh]
$ touch my_script.sh

(kali㉿kali)-[~/Kamalesh]
$ ls
my_script.sh
```

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

bash

#!/bin/bash

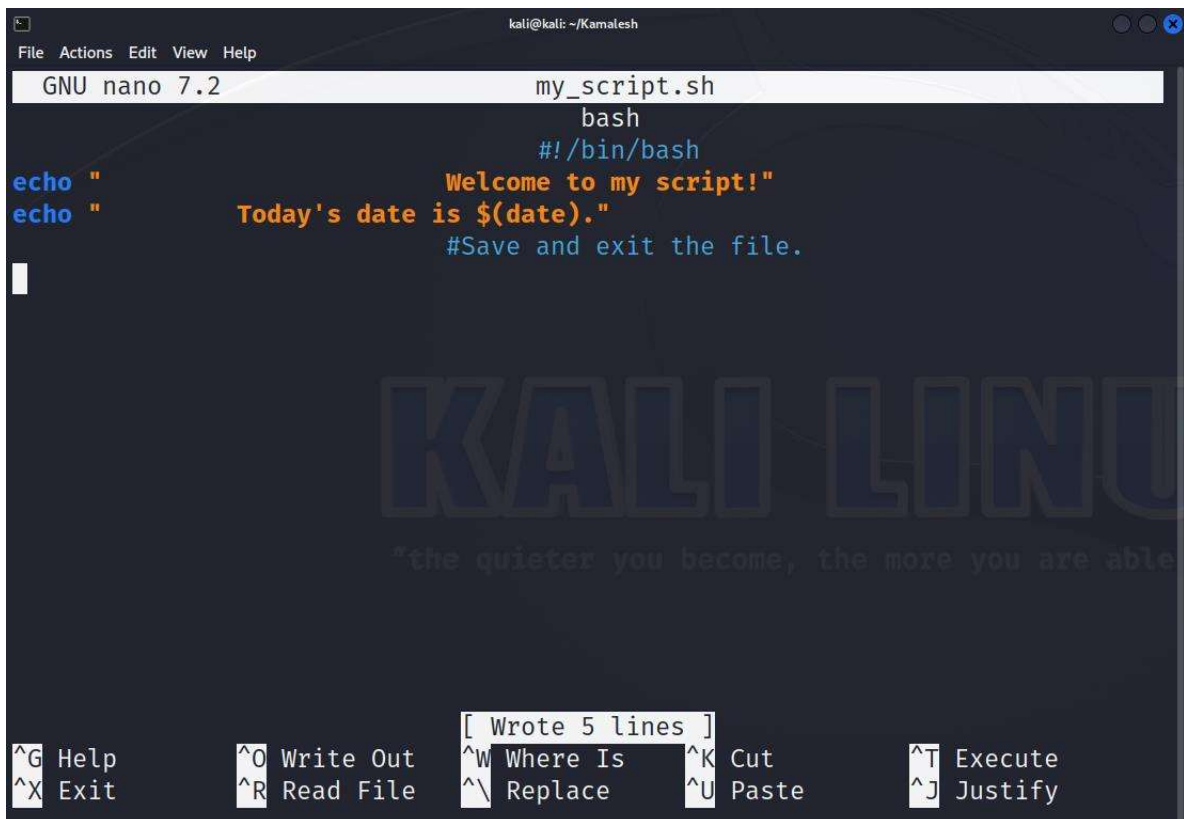
echo "Welcome to my script!"

echo "Today's date is \$(date)."

Save and exit the file.

Note:

I've just added the above lines as specified and I've not considered the phrase 'Save and exit file' as an Output, but as an instruction for the assignment alone.

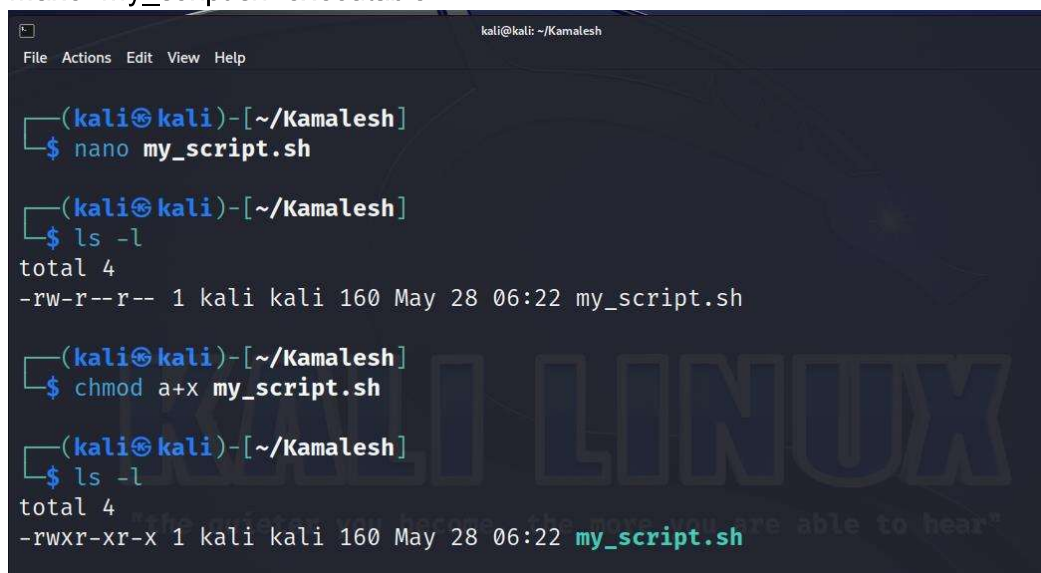


```
kali@kali: ~/Kamalesh
File Actions Edit View Help
GNU nano 7.2 my_script.sh
bash
#!/bin/bash
Welcome to my script!
Today's date is $(date).
#Save and exit the file.
```

[Wrote 5 lines]

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute
^X Exit	^R Read File	^_ Replace	^U Paste	^J Justify

- Make "my_script.sh" executable.



```
kali@kali: ~/Kamalesh
File Actions Edit View Help

(kali@kali)-[~/Kamalesh]
$ nano my_script.sh

(kali@kali)-[~/Kamalesh]
$ ls -l
total 4
-rw-r--r-- 1 kali kali 160 May 28 06:22 my_script.sh

(kali@kali)-[~/Kamalesh]
$ chmod a+x my_script.sh

(kali@kali)-[~/Kamalesh]
$ ls -l
total 4
-rwxr-xr-x 1 kali kali 160 May 28 06:22 my_script.sh
```

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

A terminal window with a dark background and light blue text. The window title is 'kali@kali: ~/Kamalesh'. The menu bar shows 'File', 'Actions', 'Edit', 'View', and 'Help'. The terminal shows a prompt '(kali@kali)-[~/Kamalesh]' followed by the command '\$ sh my_script.sh'. After another prompt '(kali@kali)-[~/Kamalesh]', the command '\$ exit' is entered. The output of the script is displayed: 'Welcome to my script!' followed by 'Today's date is Sun May 28 06:35:11 AM EDT 2023.' on the next line.

```
kali@kali: ~/Kamalesh
File Actions Edit View Help

(kali@kali)-[~/Kamalesh]
$ sh my_script.sh
(kali@kali)-[~/Kamalesh]
$ exit
exit

Welcome to my script!
Today's date is Sun May 28 06:35:11 AM EDT 2023.
```

Observation:

'Bash' runs first, and the shell enters Bash entry.

By exiting Bash by using 'exit' the rest of the script is executed that prints the specified content, including the \$(date) command.

Task 3: Command Execution and Pipelines

- List all the processes running on your system using the "ps" command.
(\$ ps aux – to display all process running on my system)

```

kali@kali: ~/Kamalesh
File Actions Edit View Help

(kali@kali)~$ ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.3 167740 12160 ?        Ss   04:59   0:02 /sbin/init
root         2  0.0  0.0      0     0 ?        S    04:59   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        I<   04:59   0:00 [rcu_gp]
root         4  0.0  0.0      0     0 ?        I<   04:59   0:00 [rcu_par_g
root         5  0.0  0.0      0     0 ?        I<   04:59   0:00 [slub_flus
root         6  0.0  0.0      0     0 ?        I<   04:59   0:00 [netns]
root        10  0.0  0.0      0     0 ?        I<   04:59   0:00 [mm_percpu
root        11  0.0  0.0      0     0 ?        I    04:59   0:00 [rcu_tasks
root        12  0.0  0.0      0     0 ?        I    04:59   0:00 [rcu_tasks
root        13  0.0  0.0      0     0 ?        I    04:59   0:00 [rcu_tasks
root        14  0.0  0.0      0     0 ?        S    04:59   0:00 [ksoftirqd
root        15  0.0  0.0      0     0 ?        I    04:59   0:05 [rcu_preem
root        16  0.0  0.0      0     0 ?        S    04:59   0:00 [migration
root        18  0.0  0.0      0     0 ?        S    04:59   0:00 [cpuhp/0]
root        19  0.0  0.0      0     0 ?        S    04:59   0:00 [cpuhp/1]
root        20  0.0  0.0      0     0 ?        S    04:59   0:00 [migration
root        21  0.0  0.0      0     0 ?        S    04:59   0:00 [ksoftirqd
root        23  0.0  0.0      0     0 ?        I<   04:59   0:00 [kworker/1
root        24  0.0  0.0      0     0 ?        S    04:59   0:00 [cpuhp/2]
root        25  0.0  0.0      0     0 ?        S    04:59   0:00 [migration
root        26  0.0  0.0      0     0 ?        S    04:59   0:00 [ksoftirqd
root        28  0.0  0.0      0     0 ?        I<   04:59   0:00 [kworker/2

```

- Use the "grep" command to filter the processes list and display only the processes with "bash" in their name.
(used 'pipe |' to connect inputs and outputs of the specified commands.)

```

kali@kali: ~/Kamalesh
File Actions Edit View Help

(kali@kali)~$ ps -ef | grep -i bash
kali      59131  39789  0 06:54 pts/0    00:00:00 grep --color=auto -i bash

```

Note:

I've tried ps commands with multiple options such as 'aux', '-e', 'x' etc. And no option let the ps command to display processes with name 'bash'

So, I executed anyway, and the 'grep' command found only one match, which is the command that I used to search for bash itself.

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

```

kali@kali: ~/Kamalesh
File Actions Edit View Help

(kali@kali)~$ ps -ef | grep -i bash
kali      61383  39789  0 06:59 pts/0    00:00:00 grep --color=auto -i bash

(kali@kali)~$ ps -ef | grep -i bash | wc -l
1

```