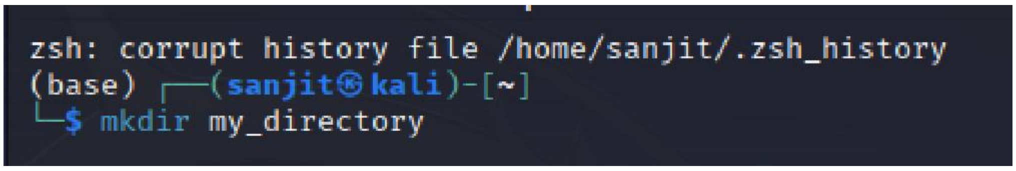
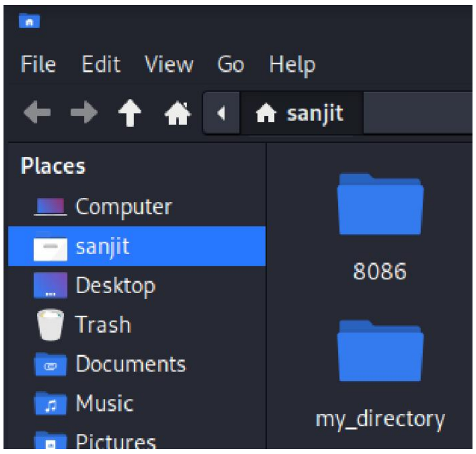
SANJIT NARAYANAN G
20BCE0052
CYBER SECURITY AND ETHICAL HACKING ASSIGNMENT - 2

TASK 1: FILE AND DIRECTORY MANIPULATION

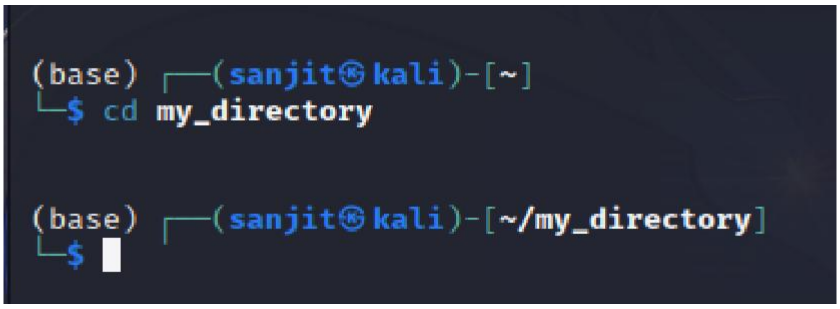
1. Create a directory called "my_directory":



```
zsh: corrupt history file /home/sanjit/.zsh_history
(base) ┌──(sanjit㉿kali)-[~]
└─$ mkdir my_directory
```

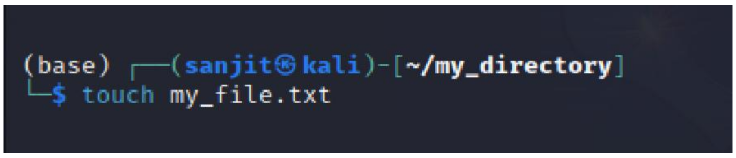


2. Navigate into the "my_directory":

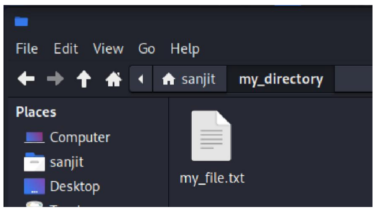


```
(base) ┌──(sanjit㉿kali)-[~]
└─$ cd my_directory

(base) ┌──(sanjit㉿kali)-[~/my_directory]
└─$ 
```

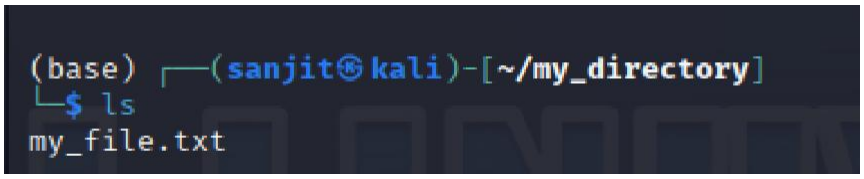
3. Create an empty file called "my_file.txt":



```
(base) ┌──(sanjit㉿kali)-[~/my_directory]
└─$ touch my_file.txt
```

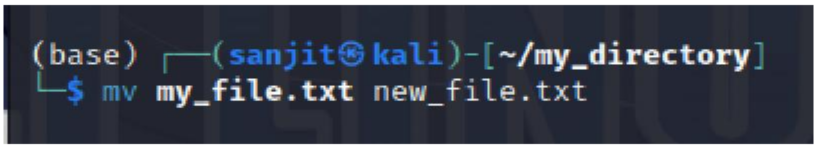


4. List all the files and directories in the current directory:

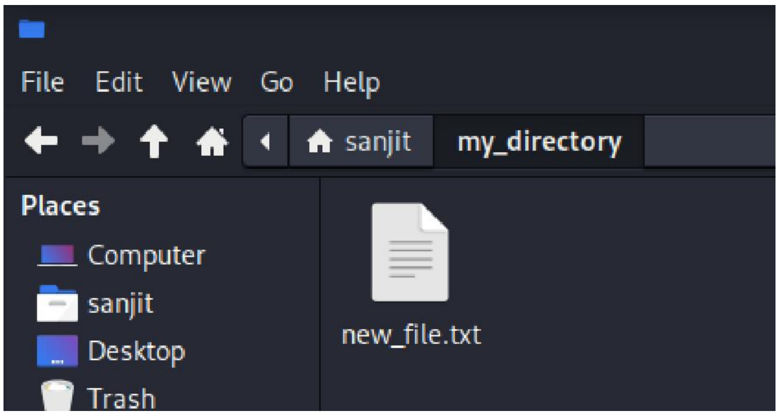


```
(base) ┌──(sanjit㉿kali)-[~/my_directory]
└─$ ls
my_file.txt
```

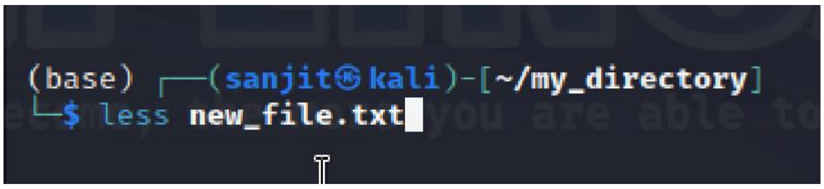
5. Rename "my_file.txt" to "new_file.txt":



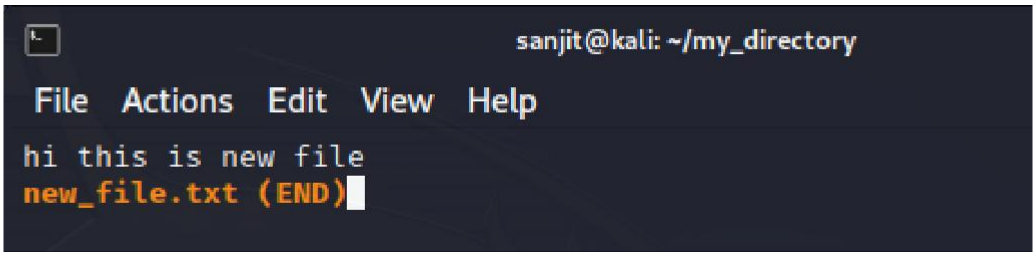
```
(base) ┌──(sanjit㉿kali)-[~/my_directory]
└─$ mv my_file.txt new_file.txt
```



6. Display the content of "new_file.txt" using a pager tool (e.g., less):

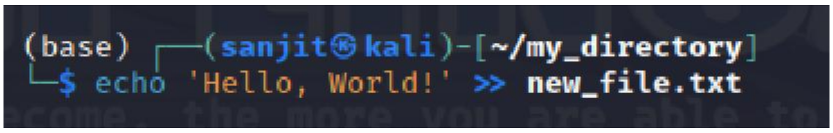


```
(base) ┌──(sanjit㉿kali)-[~/my_directory]
└─$ less new_file.txt
```

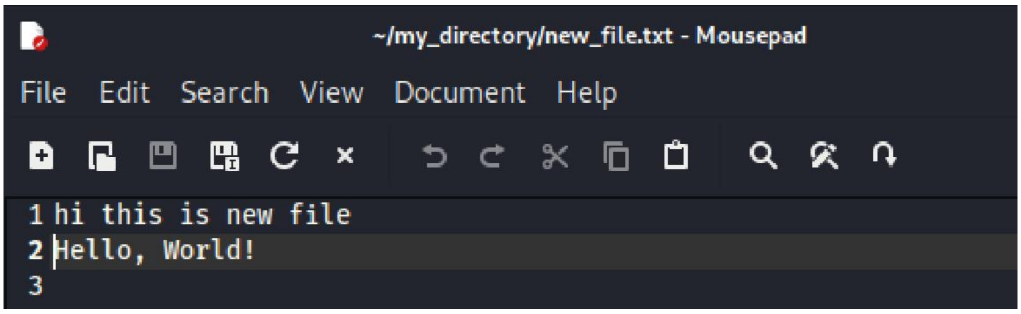


```
hi this is new file
new_file.txt (END)
```

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



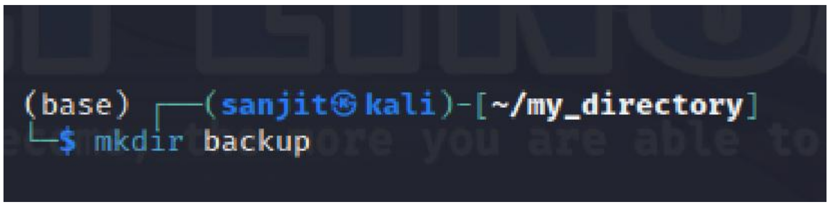
```
(base) ┌──(sanjit㉿kali)-[~/my_directory]
└─$ echo 'Hello, World!' >> new_file.txt
```



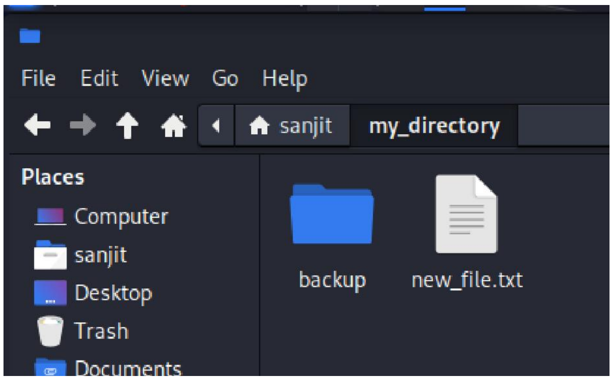
~/my_directory/new_file.txt - Mousepad
File Edit Search View Document Help

```
hi this is new file
Hello, World!

```

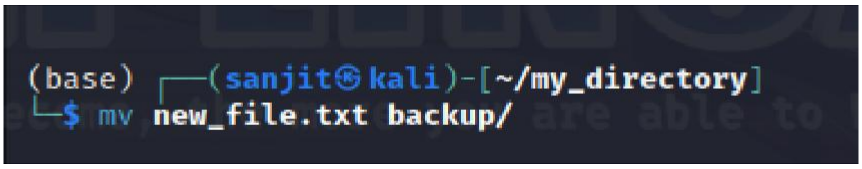
8. Create a new directory called "backup" within "my_directory":



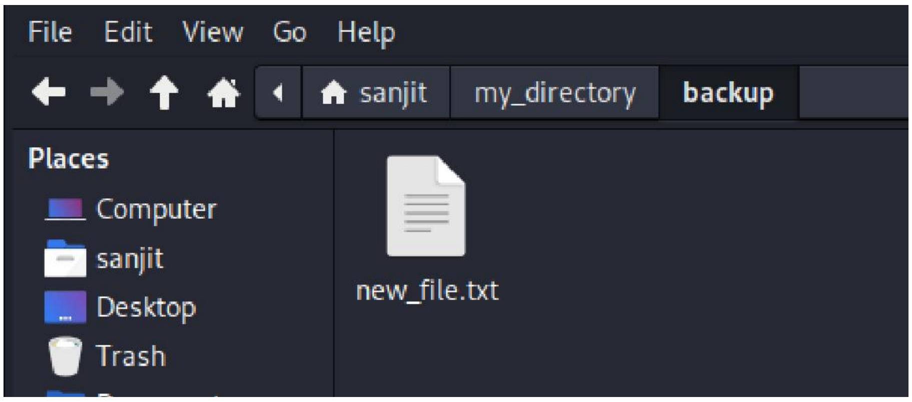
```
(base) ┌──(sanjit㉿kali)-[~/my_directory]
└─$ mkdir backup
```



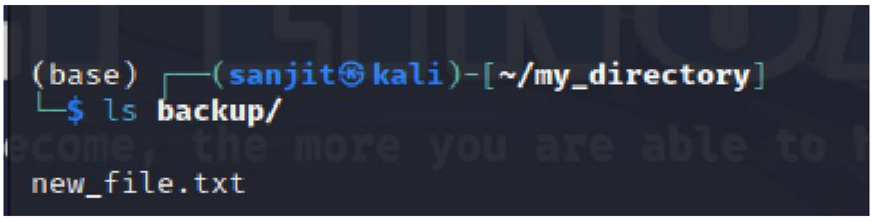
9. Move "new_file.txt" to the "backup" directory:



```
(base) ┌──(sanjit㉿kali)-[~/my_directory]
└─$ mv new_file.txt backup/
```

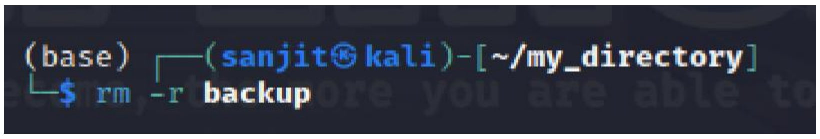


10. Verify that "new_file.txt" is now located in the "backup" directory:



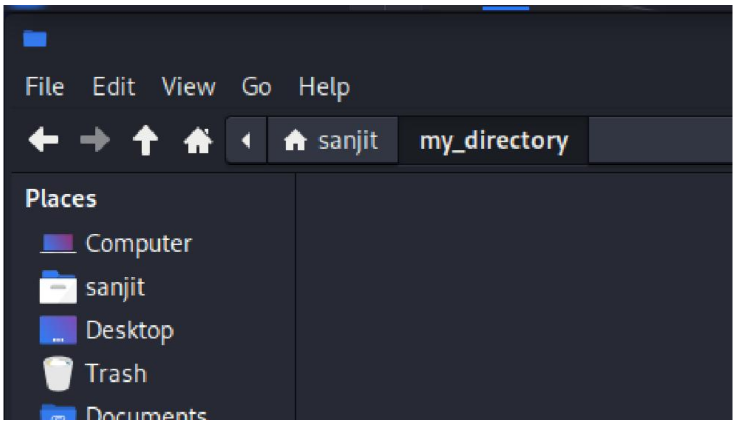
```
(base) ┌──(sanjit㉿kali)-[~/my_directory]
└─$ ls backup/

new_file.txt
```

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



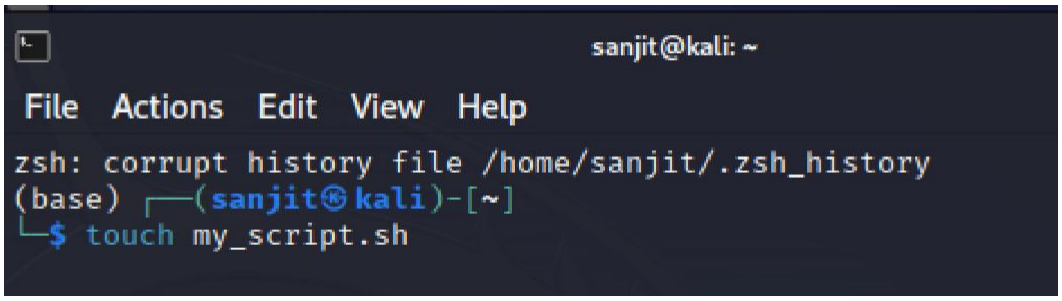
```
(base) ┌──(sanjit㉿kali)-[~/my_directory]
└─$ rm -r backup
```

YOU CAN SEE THAT my_directory IS EMPTY.

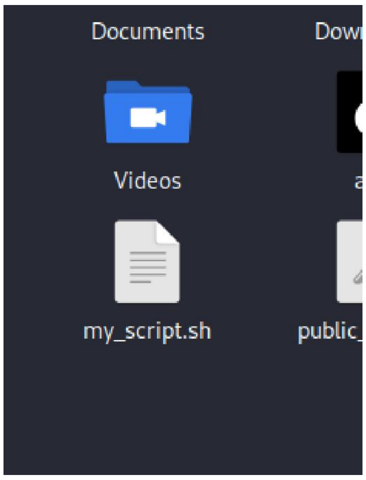


TASK 2: PERMISSIONS AND SCRIPTING

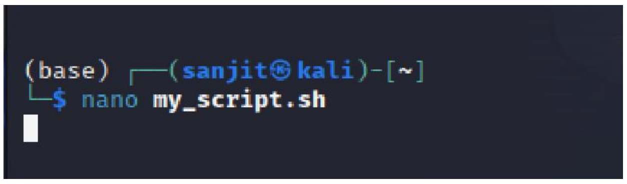
1. Create a new file called "my_script.sh":



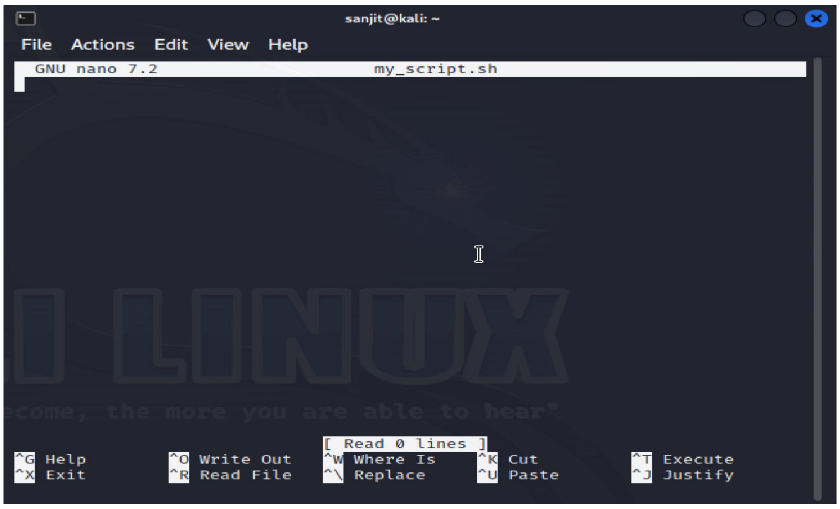
```
zsh: corrupt history file /home/sanjit/.zsh_history
(base) ┌──(sanjit㉿kali)-[~]
└─$ touch my_script.sh
```



2. Edit "my_script.sh" using a text editor (e.g., nano):

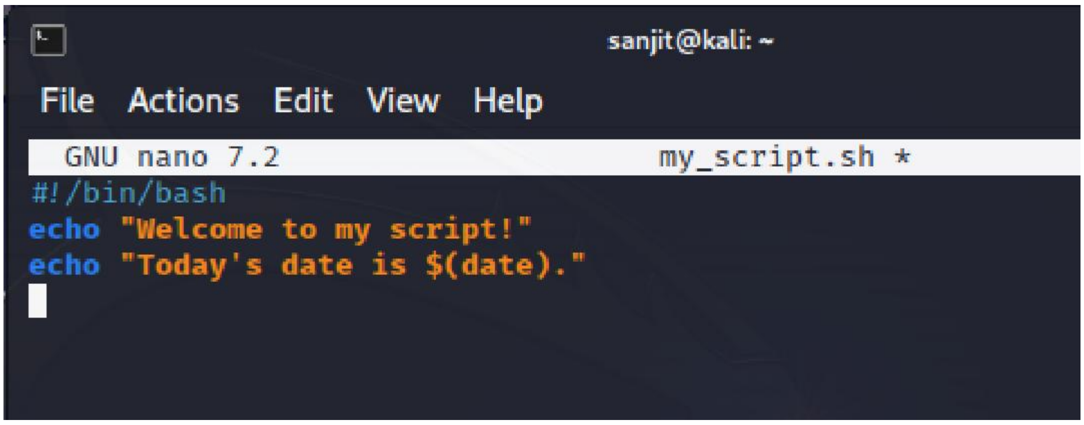


```
(base) ┌──(sanjit㉿kali)-[~]
└─$ nano my_script.sh
```



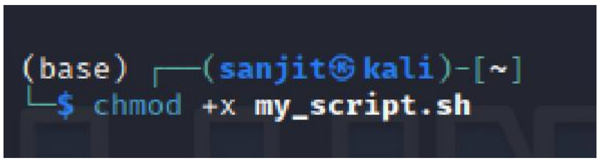
```
  GNU nano 7.2                    my_script.sh

                          [ Read 0 lines ]
^G Help       ^O Write Out   ^W Where Is    ^K Cut         ^T Execute
^X Exit       ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```

3. In the text editor, add the following lines to "my_script.sh":



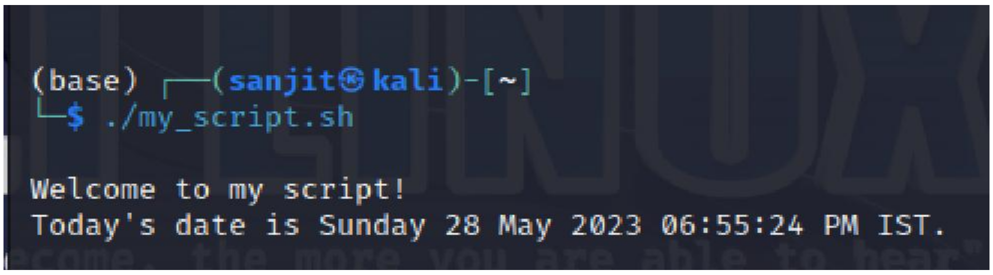
```
  GNU nano 7.2                          my_script.sh *
#!/bin/bash
echo "Welcome to my script!"
echo "Today's date is $(date)."
```

4. Make "my_script.sh" executable:



```
(base) ┌──(sanjit㉿kali)-[~]
└─$ chmod +x my_script.sh
```

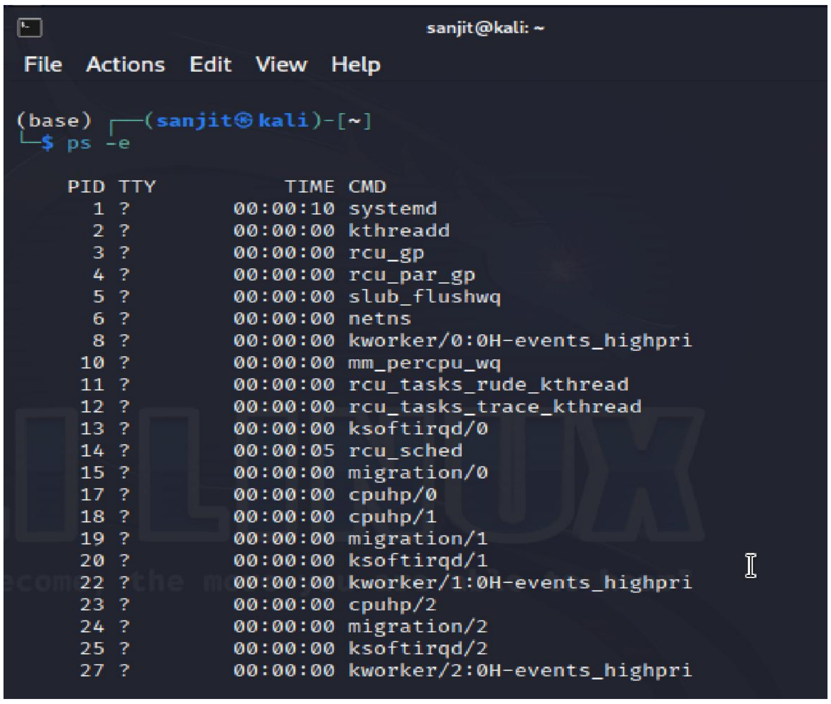
5. Run "my_script.sh" and verify the output:



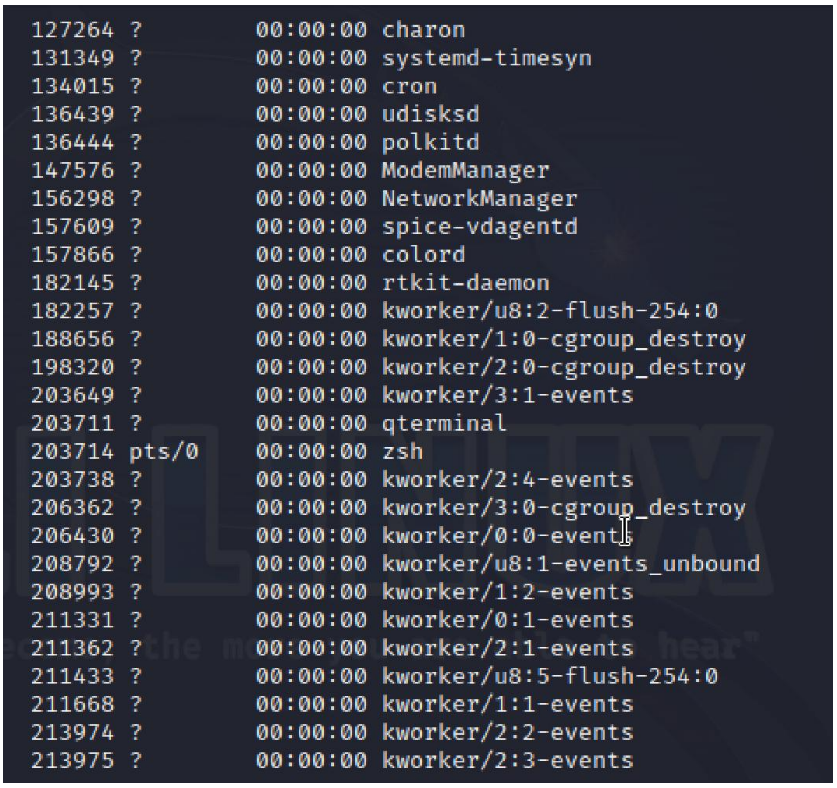
```
(base) ┌──(sanjit㉿kali)-[~]
└─$ ./my_script.sh

Welcome to my script!
Today's date is Sunday 28 May 2023 06:55:24 PM IST.
```

TASK 3: COMMAND EXECUTION AND PIPELINES

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

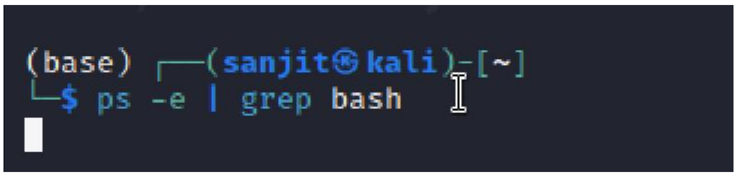


```
(base) ┌──(sanjit㉿kali)-[~]
└─$ ps -e

    PID TTY          TIME CMD
      1 ?        00:00:10 systemd
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00 rcu_gp
      4 ?        00:00:00 rcu_par_gp
      5 ?        00:00:00 slub_flushwq
      6 ?        00:00:00 netns
      8 ?        00:00:00 kworker/0:0H-events_highpri
     10 ?        00:00:00 mm_percpu_wq
     11 ?        00:00:00 rcu_tasks_rude_kthread
     12 ?        00:00:00 rcu_tasks_trace_kthread
     13 ?        00:00:00 ksoftirqd/0
     14 ?        00:00:05 rcu_sched
     15 ?        00:00:00 migration/0
     17 ?        00:00:00 cpuhp/0
     18 ?        00:00:00 cpuhp/1
     19 ?        00:00:00 migration/1
     20 ?        00:00:00 ksoftirqd/1
     22 ?        00:00:00 kworker/1:0H-events_highpri
     23 ?        00:00:00 cpuhp/2
     24 ?        00:00:00 migration/2
     25 ?        00:00:00 ksoftirqd/2
     27 ?        00:00:00 kworker/2:0H-events_highpri
```



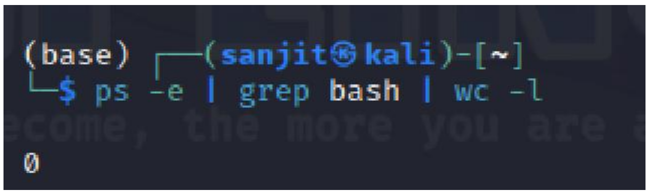
```
 127264 ?        00:00:00 charon
 131349 ?        00:00:00 systemd-timesyn
 134015 ?        00:00:00 cron
 136439 ?        00:00:00 udisksd
 136444 ?        00:00:00 polkitd
 147576 ?        00:00:00 ModemManager
 156298 ?        00:00:00 NetworkManager
 157609 ?        00:00:00 spice-vdagentd
 157866 ?        00:00:00 colord
 182145 ?        00:00:00 rtkit-daemon
 182257 ?        00:00:00 kworker/u8:2-flush-254:0
 188656 ?        00:00:00 kworker/1:0-cgroup_destroy
 198320 ?        00:00:00 kworker/2:0-cgroup_destroy
 203649 ?        00:00:00 kworker/3:1-events
 203711 ?        00:00:00 qterminal
 203714 pts/0    00:00:00 zsh
 203738 ?        00:00:00 kworker/2:4-events
 206362 ?        00:00:00 kworker/3:0-cgroup_destroy
 206430 ?        00:00:00 kworker/0:0-events
 208792 ?        00:00:00 kworker/u8:1-events_unbound
 208993 ?        00:00:00 kworker/1:2-events
 211331 ?        00:00:00 kworker/0:1-events
 211362 ?        00:00:00 kworker/2:1-events
 211433 ?        00:00:00 kworker/u8:5-flush-254:0
 211668 ?        00:00:00 kworker/1:1-events
 213974 ?        00:00:00 kworker/2:2-events
 213975 ?        00:00:00 kworker/2:3-events
```

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



```
(base) ┌──(sanjit㉿kali)-[~]
└─$ ps -e | grep bash
```

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



```
(base) ┌──(sanjit㉿kali)-[~]
└─$ ps -e | grep bash | wc -l
0
```