

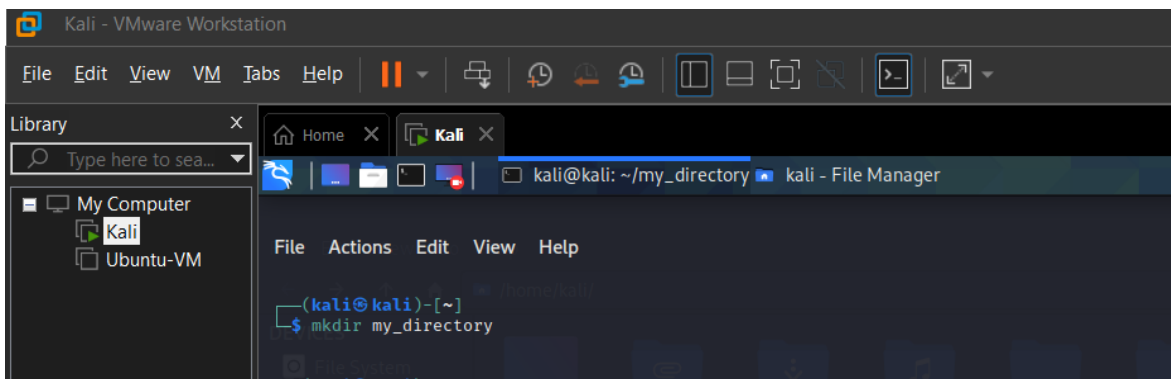
Darshan Jain

20BCE2657

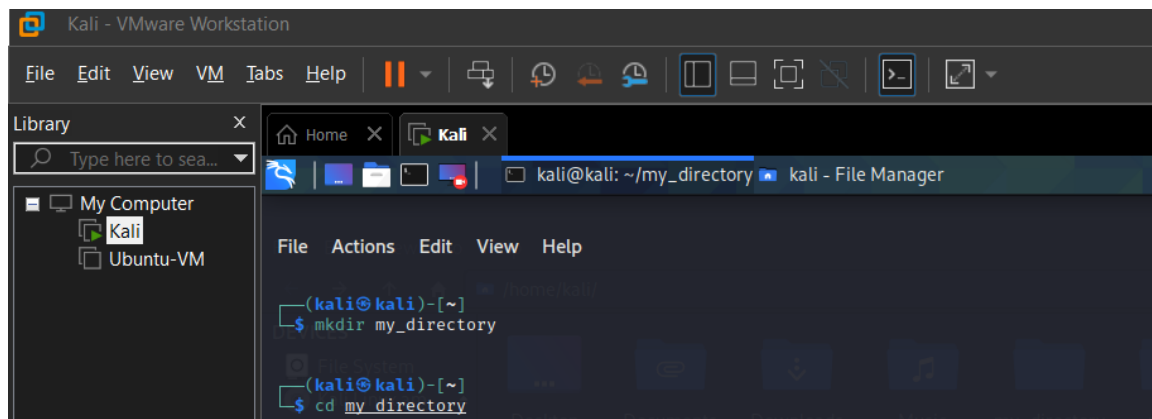
Assignment: Bash Shell Basics

Task 1: File and Directory Manipulation

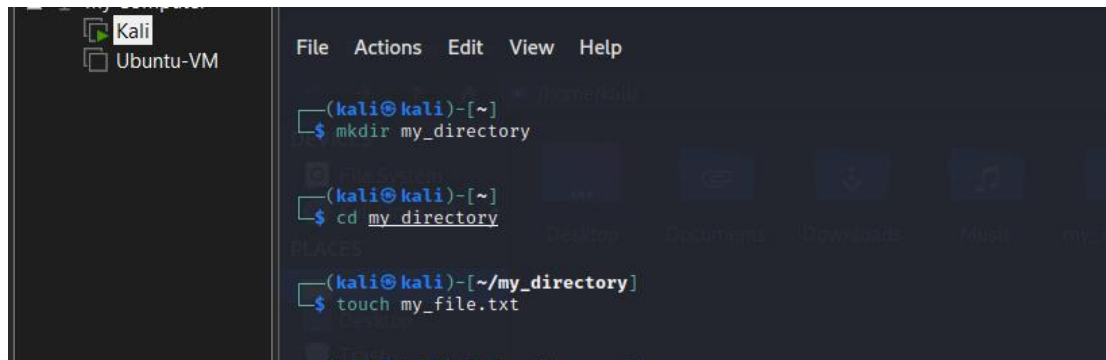
1. Create a directory called "my_directory".



2. Navigate into the "my_directory".



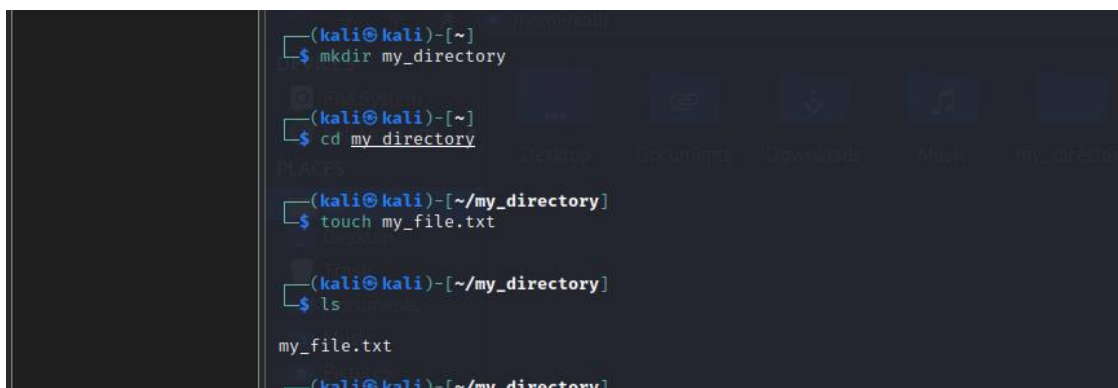
3. Create an empty file called "my_file.txt".



A terminal window with a dark background. The left sidebar shows a file manager with 'Kali' and 'Ubuntu-VM' folders. The terminal has a menu bar with 'File', 'Actions', 'Edit', 'View', and 'Help'. The prompt is '(kali@kali)-[~]'. The user enters '\$ mkdir my_directory'. The prompt changes to '(kali@kali)-[~]'. The user enters '\$ cd my_directory'. The prompt changes to '(kali@kali)-[~/my_directory]'. The user enters '\$ touch my_file.txt'.

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

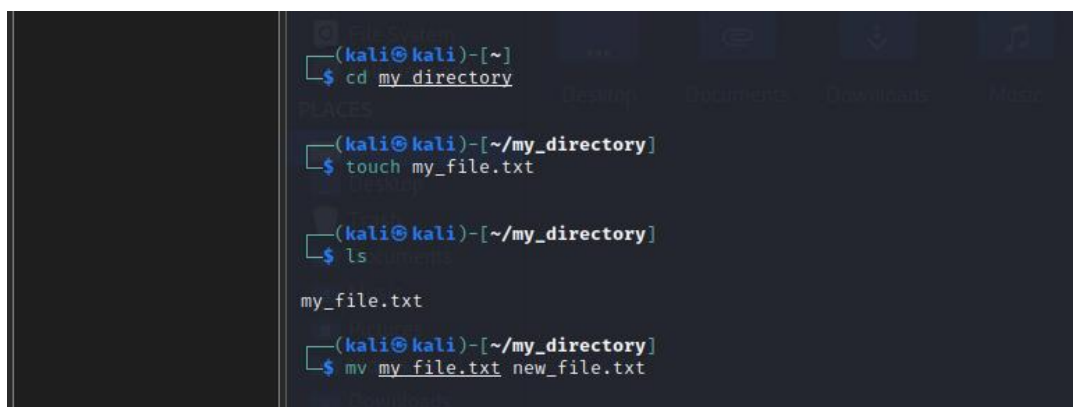
4. List all the files and directories in the current directory.



A terminal window showing the next steps. The prompt is '(kali@kali)-[~/my_directory]'. The user enters '\$ touch my_file.txt'. The prompt changes to '(kali@kali)-[~/my_directory]'. The user enters '\$ ls'. The output is 'my_file.txt'. The prompt changes to '(kali@kali)-[~/my_directory]'.

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

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



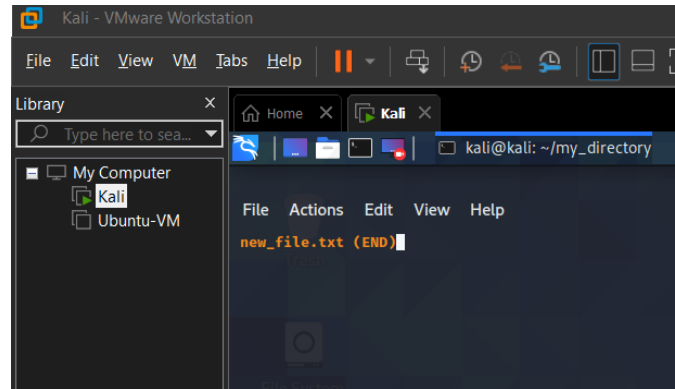
A terminal window showing the next steps. The prompt is '(kali@kali)-[~/my_directory]'. The user enters '\$ cd my_directory'. The prompt changes to '(kali@kali)-[~/my_directory]'. The user enters '\$ touch my_file.txt'. The prompt changes to '(kali@kali)-[~/my_directory]'. The user enters '\$ ls'. The output is 'my_file.txt'. The prompt changes to '(kali@kali)-[~/my_directory]'. The user enters '\$ mv my_file.txt new_file.txt'.

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

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

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



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

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

(kali㉿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".

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

dquote> "
Hello, World >> new_file.txt

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

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

```
dquote> "  
Hello, World >> new_file.txt  
  
(kali㉿kali)-[~/my_directory]  
$ mkdir backup  
  
(kali㉿kali)-[~/my_directory]  
$ mv new_file.txt backup/
```

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

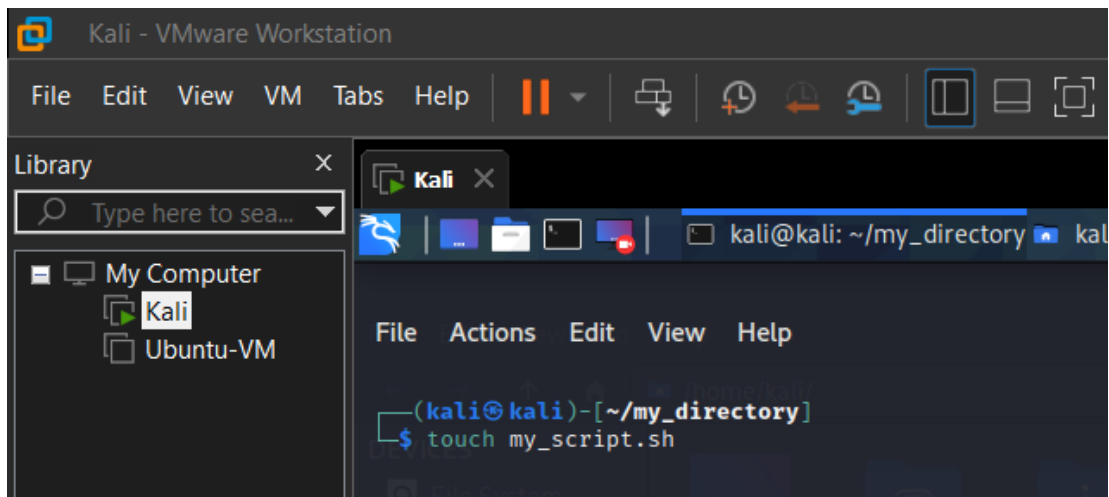
```
(kali㉿kali)-[~/my_directory]  
$ mv new_file.txt backup/  
  
(kali㉿kali)-[~/my_directory]  
$ ls backup/  
  
new_file.txt
```

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

```
(kali㉿kali)-[~/my_directory]  
$ ls backup/  
  
new_file.txt  
  
(kali㉿kali)-[~/my_directory]  
$ rm -r backup
```

Task 2: Permissions and Scripting

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



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

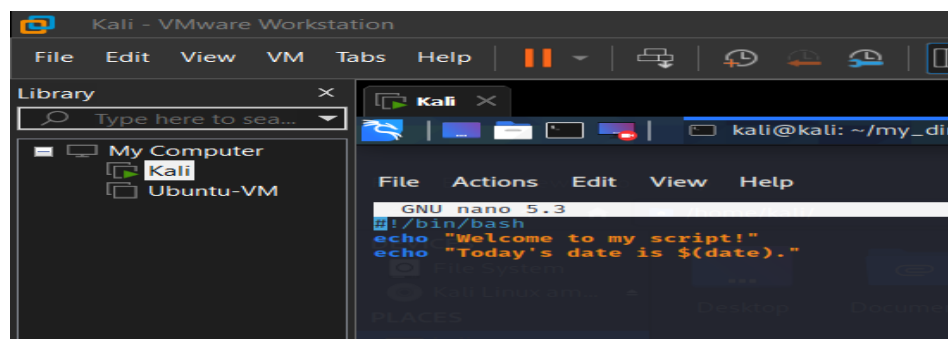
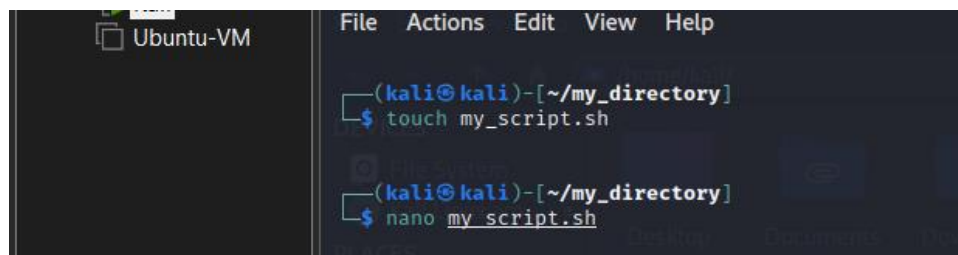
bash

#!/bin/bash

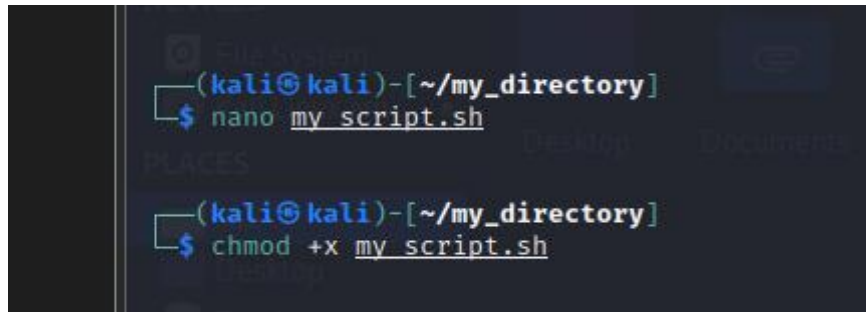
echo "Welcome to my script!"

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

Save and exit the file.



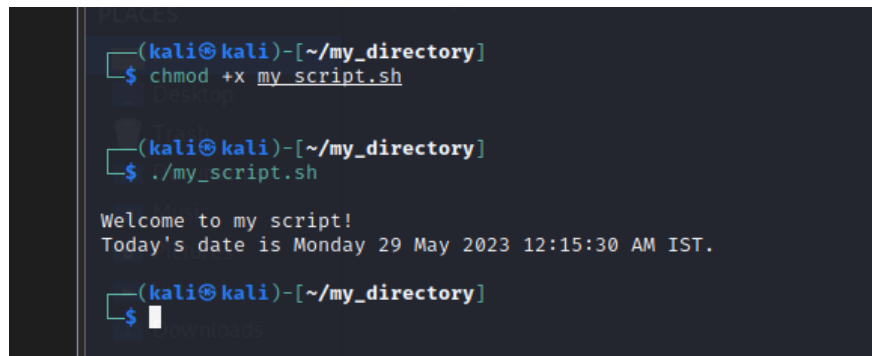
- Make "my_script.sh" executable.

A terminal window on a Kali Linux system. The prompt is (kali@kali)-[~/my_directory]. The user enters 'nano my_script.sh' and then 'chmod +x my_script.sh'.

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

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

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

A terminal window on a Kali Linux system. The prompt is (kali@kali)-[~/my_directory]. The user enters 'chmod +x my_script.sh' and then './my_script.sh'. The output of the script is displayed: 'Welcome to my script!' and 'Today's date is Monday 29 May 2023 12:15:30 AM IST.'.

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

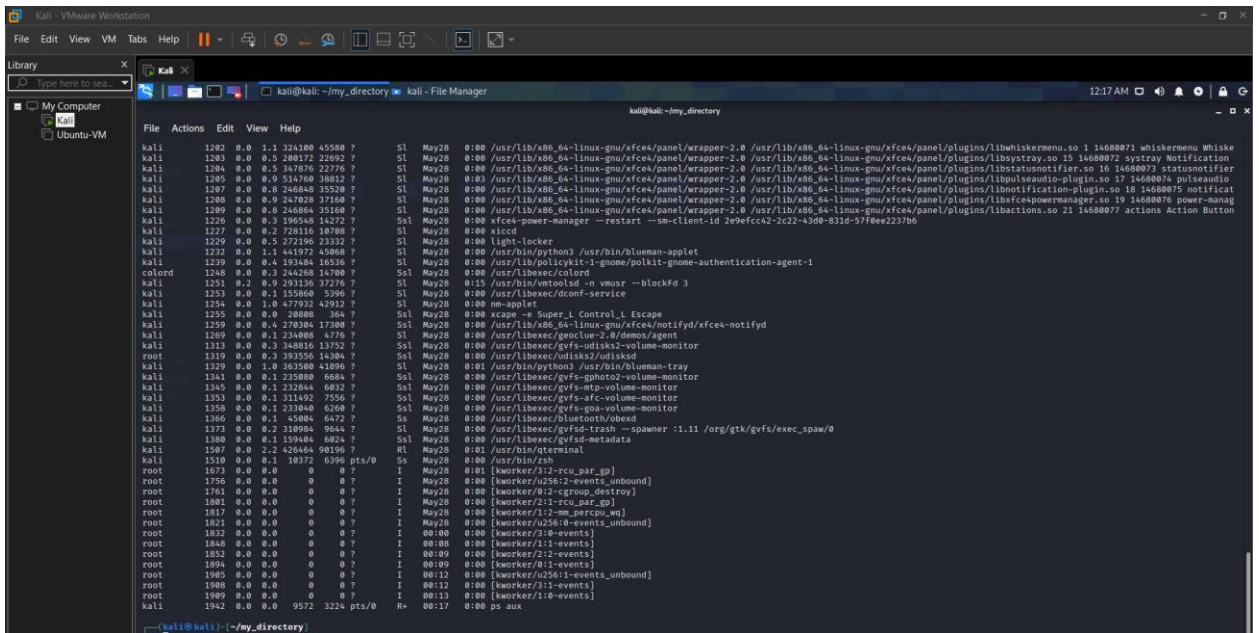
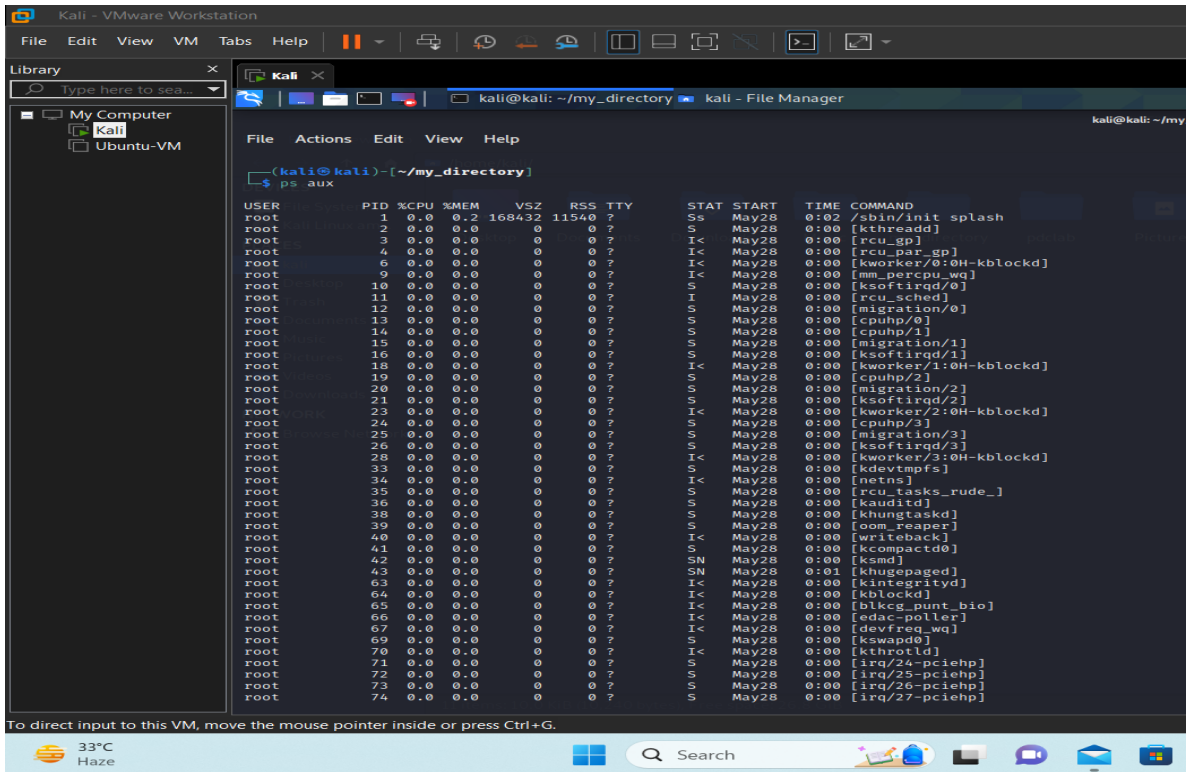
(kali@kali)-[~/my_directory]
$ ./my_script.sh

Welcome to my script!
Today's date is Monday 29 May 2023 12:15:30 AM IST.

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

Task 3: Command Execution and Pipelines

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



ps aux: The ps command is used to display information about active processes. The aux options show information for all processes, including those not attached to a terminal. This command lists all running processes on your system.

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

```

root      1761  0.0  0.0      0   0 ?      I   May28   0:00 [kworker/0:2-cgroup_destroy]
root      1801  0.0  0.0      0   0 ?      I   May28   0:00 [kworker/2:1-rcu_par_gp]
root      1817  0.0  0.0      0   0 ?      I   May28   0:00 [kworker/1:2-mm_percpu_wq]
root      1821  0.0  0.0      0   0 ?      I   May28   0:00 [kworker/u256:0-events_unbound]
root      1832  0.0  0.0      0   0 ?      I   00:00   0:00 [kworker/3:0-events]
root      1848  0.0  0.0      0   0 ?      I   00:08   0:00 [kworker/1:1-events]
root      1852  0.0  0.0      0   0 ?      I   00:09   0:00 [kworker/2:2-events]
root      1894  0.0  0.0      0   0 ?      I   00:09   0:00 [kworker/0:1-events]
root      1905  0.0  0.0      0   0 ?      I   00:12   0:00 [kworker/u256:1-events_unbound]
root      1908  0.0  0.0      0   0 ?      I   00:12   0:00 [kworker/3:1-events]
root      1909  0.0  0.0      0   0 ?      I   00:13   0:00 [kworker/1:0-events]
kali      1942  0.0  0.0    9572 3224 pts/0    R+   00:17   0:00 ps aux

(kali@kali)-[~/my_directory]
$ ps aux | grep bash

kali      1945  0.0  0.0    6112  716 pts/0    S+   00:18   0:00 grep --color=auto bash

```

grep bash: The grep command is used to search for a specific pattern or text in the given input. Here, we are searching for lines containing the word "bash" in the output of the previous ps aux command.

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

```

(kali@kali)-[~/my_directory]
$ ps aux | grep bash

kali      1945  0.0  0.0    6112  716 pts/0    S+   00:18   0:00 grep --color=auto bash

(kali@kali)-[~/my_directory]
$ ps aux | grep bash | wc -l

1

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

```

wc -l: The wc command is used to count words, lines, and characters in the given input. The -l option tells wc to count the number of lines in the input. In this case, we are counting the number of lines in the filtered output of the previous grep command, which gives us the count of processes with "bash" in their name.