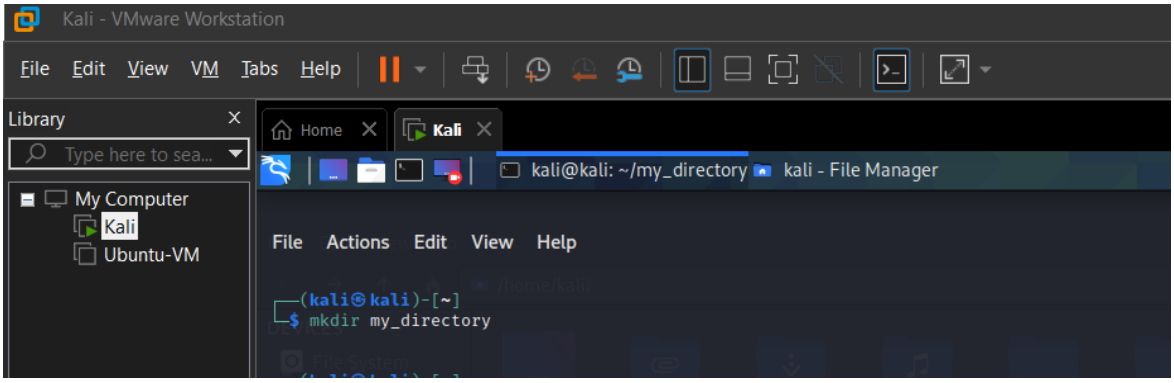
Darshan Jain
20BCE2657

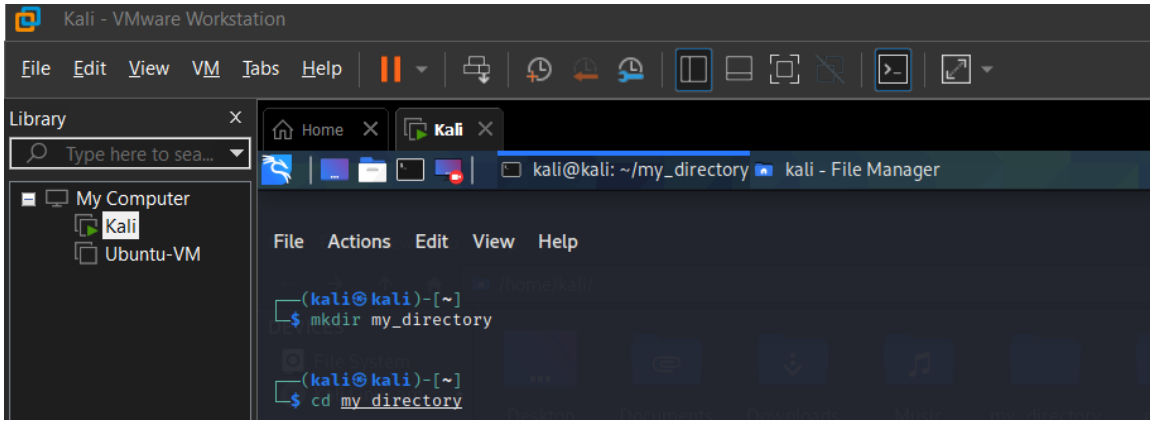
Assignment: Bash Shell Basics

Task 1: File and Directory Manipulation

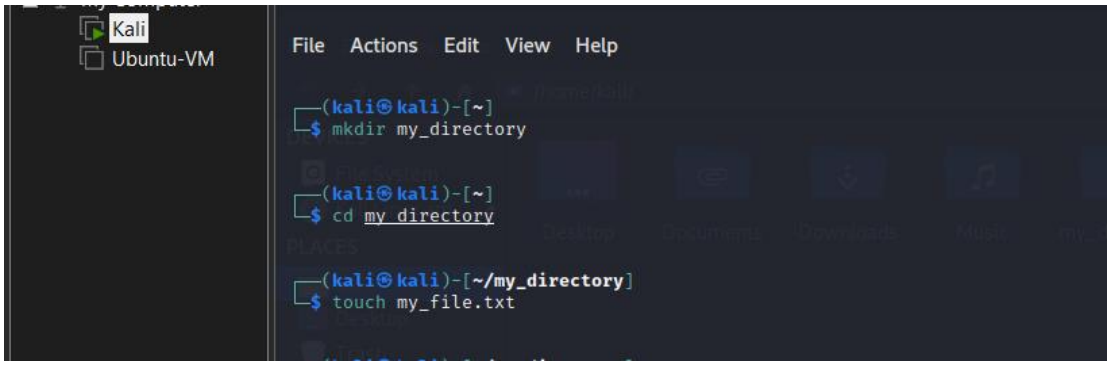
1. Create a directory called "my_directory".



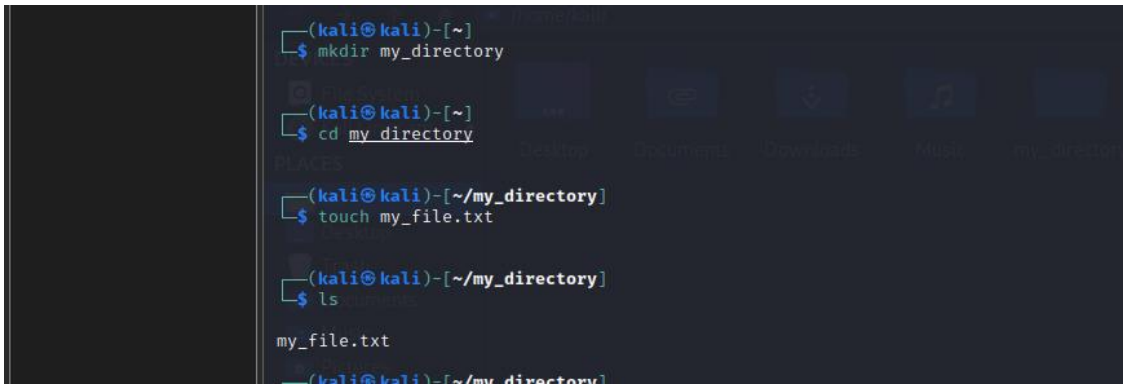
2. Navigate into the "my_directory".



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



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

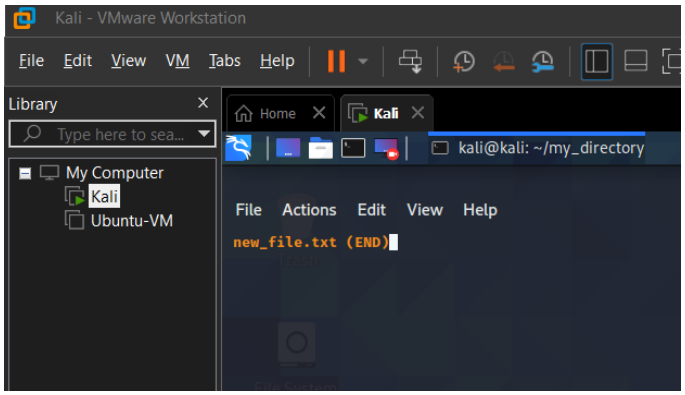


5. Rename "my_file.txt" to "new_file.txt".

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.

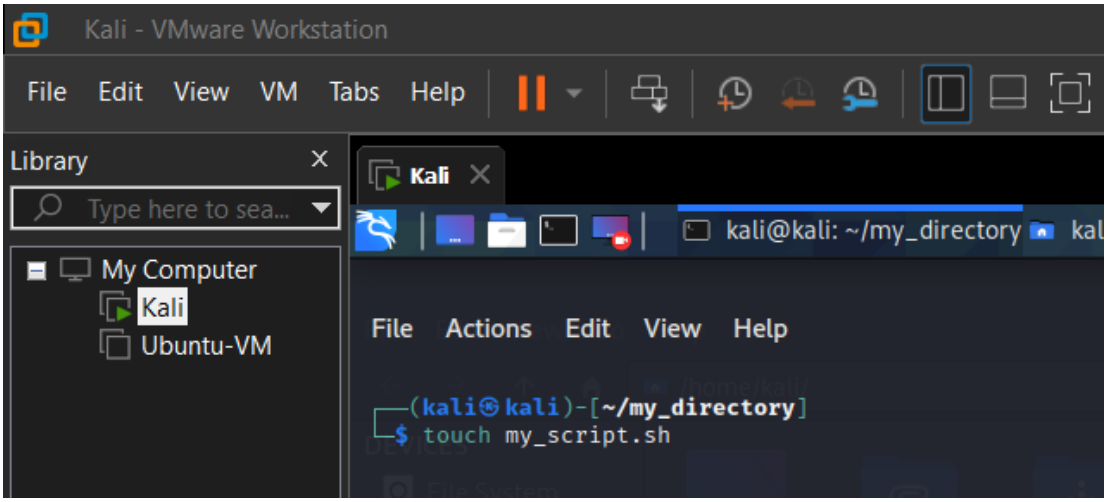
```
┌──(kali㉿kali)-[~/my_directory]
└─$ ls backup/

new_file.txt

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

Task 2: Permissions and Scripting

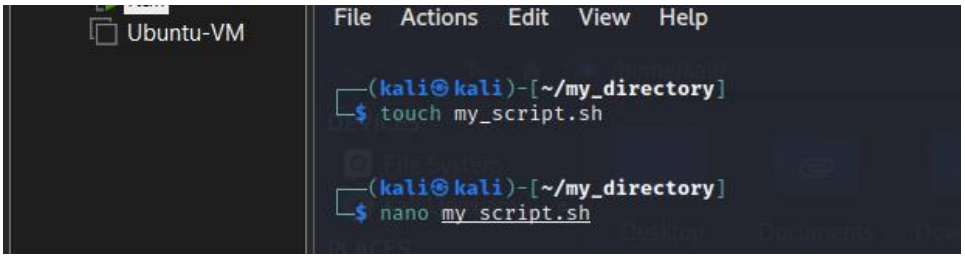
- Create a new file called "my_script.sh".

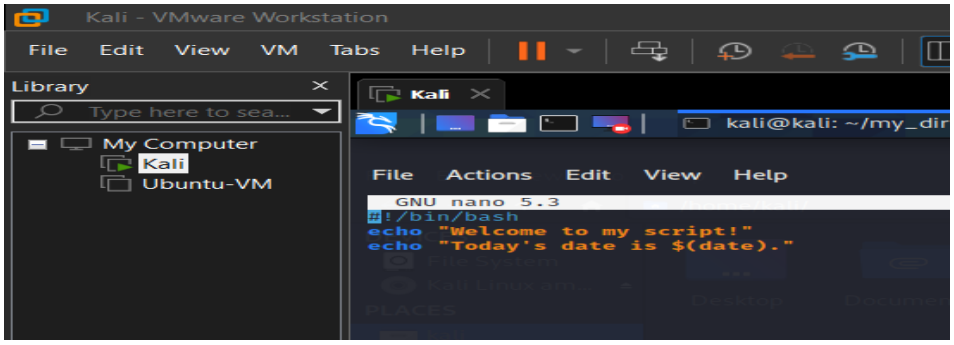


- Edit "my_script.sh" using a text editor of your choice and add the following lines:

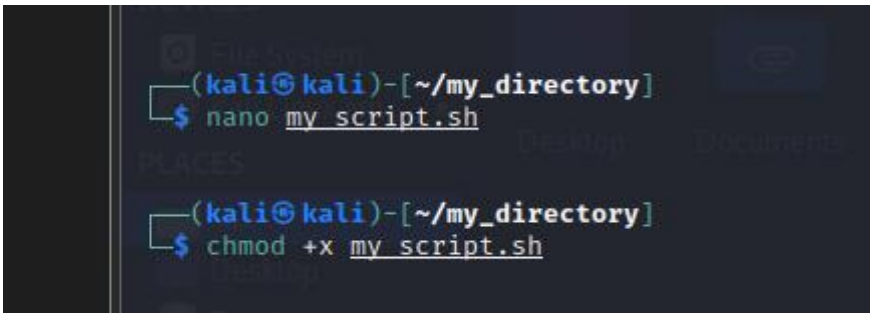
bash

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

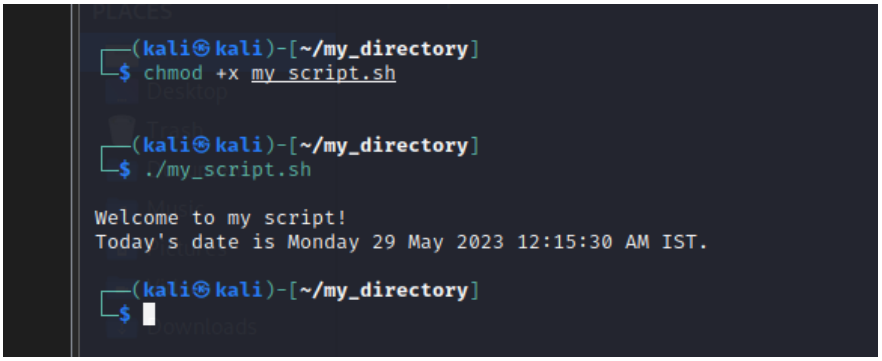




- Make "my_script.sh" executable.

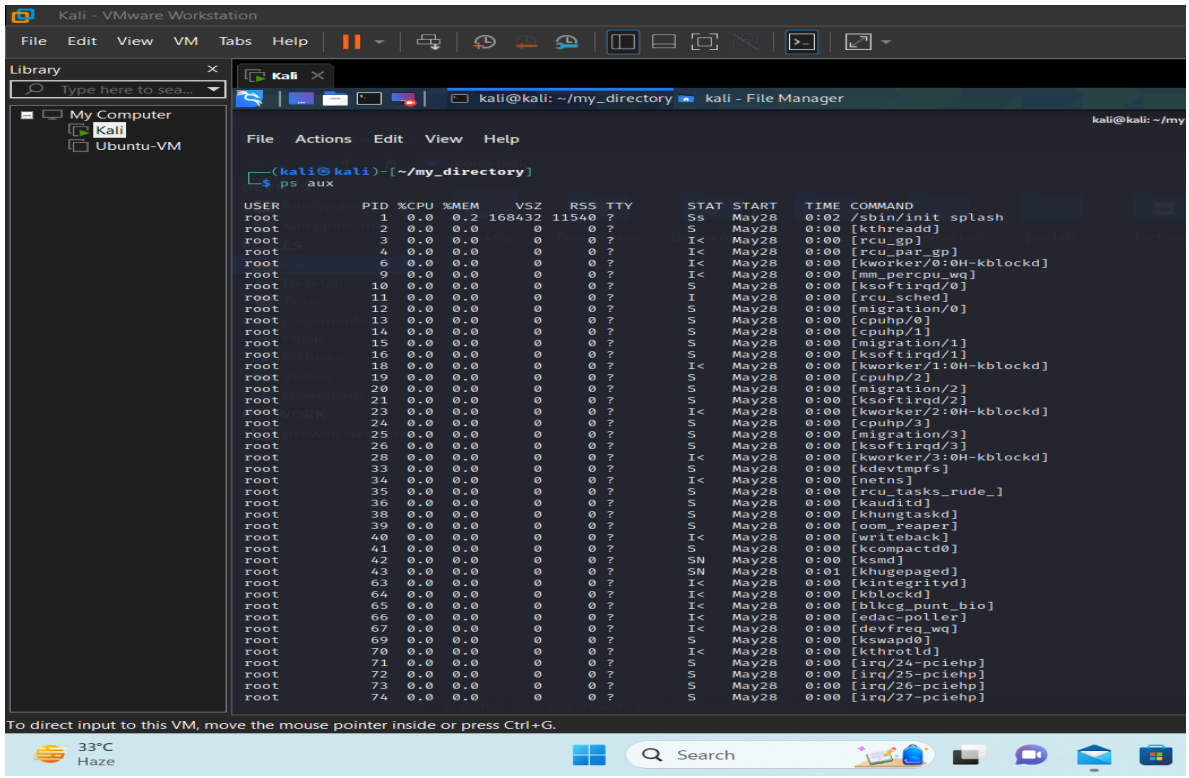


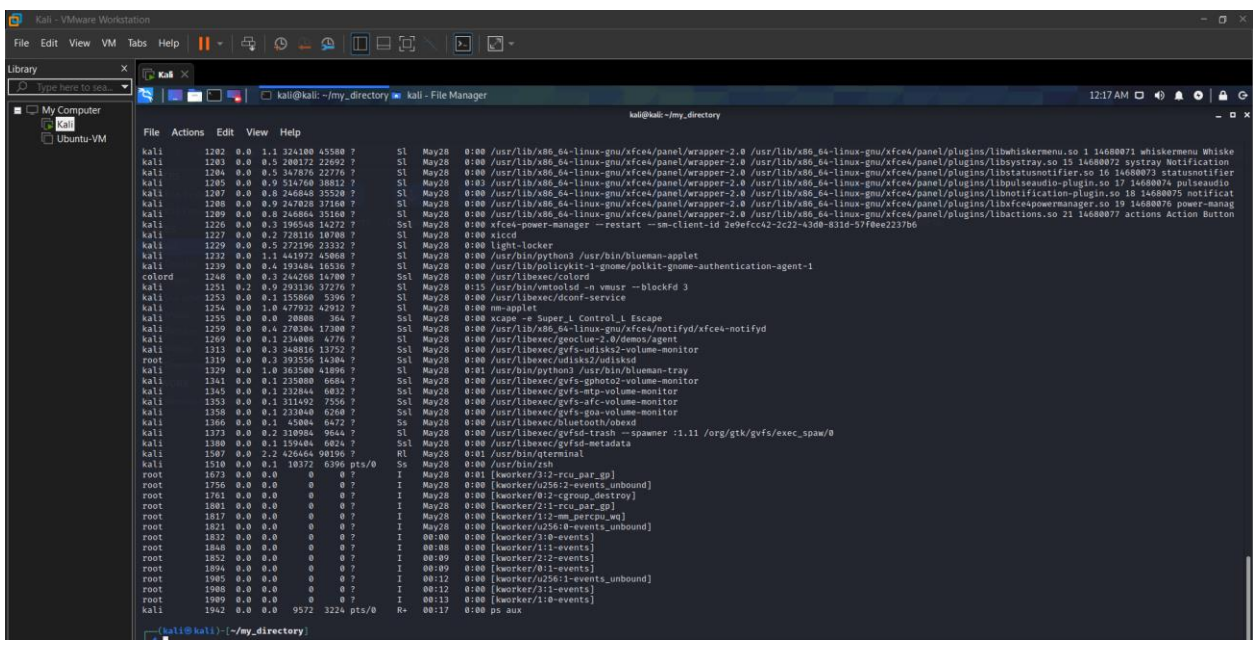
- Run "my_script.sh" and verify that the output matches the expected result.



Task 3: Command Execution and Pipelines

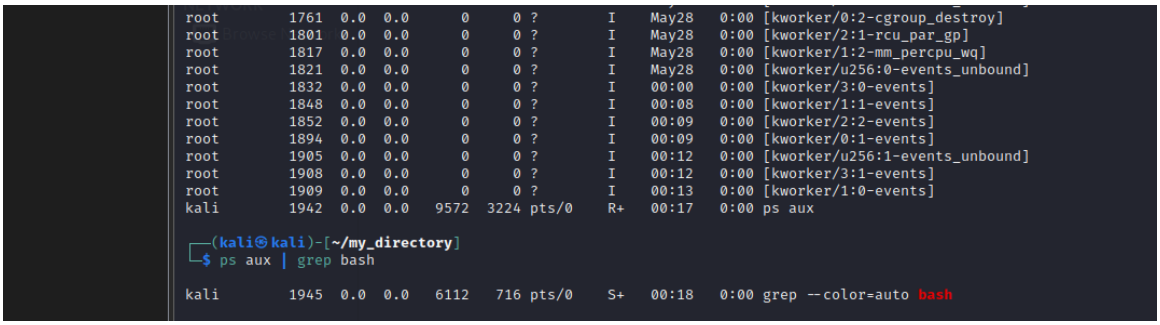
- List all the processes running on your system using the "ps" command.





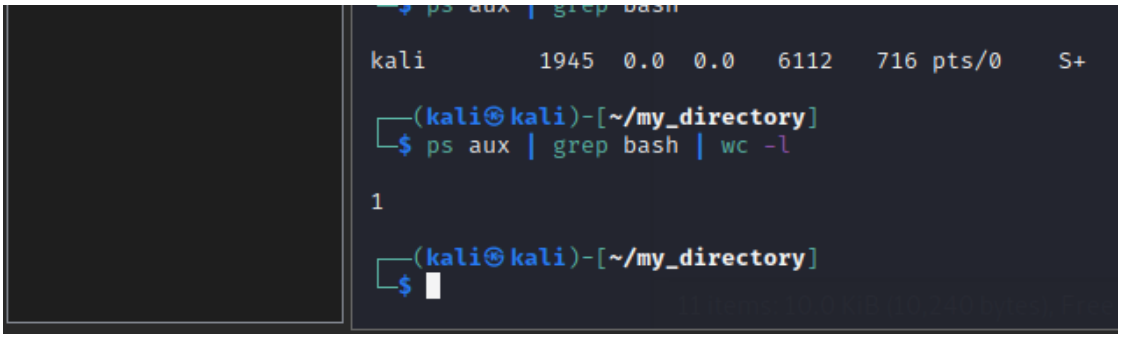
ps aux: The ps command is used to display information about active processes. The aux options show information for all processes, including those not attached to a terminal. This command lists all running processes on your system.

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



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.



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.