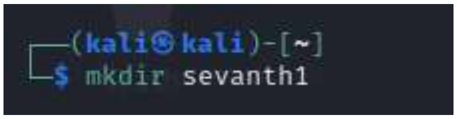
Assignment 2: Bash Shell Basics

<u>**NAME : T. SEVANTH SRI NAGA RAJA**</u>
REG NO. : 20BCR7080
CAMPUS : VIT-AP

Task 1: File and Directory Manipulation

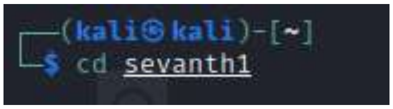
1. **Create a directory called "my_directory".**



```
┌──(kali㉿kali)-[~]
└─$ mkdir sevanth1
```

This command creates a new directory named "my_directory" in the current working directory.

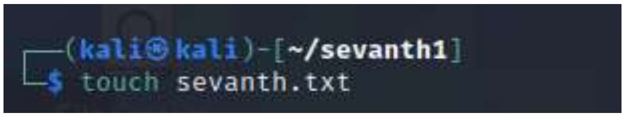
2. **Navigate into the "my_directory".**



```
┌──(kali㉿kali)-[~]
└─$ cd sevanth1
```

This command changes the current working directory to "my_directory".

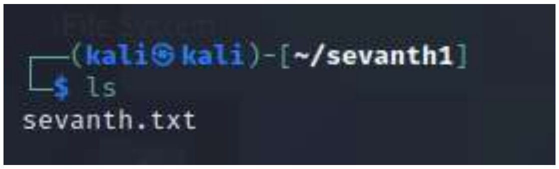
3. **Create an empty file called "my_file.txt".**



```
┌──(kali㉿kali)-[~/sevanth1]
└─$ touch sevanth.txt
```

The **touch** command is used to create an empty file. In this case, it creates a file named "my_file.txt" in the current directory.

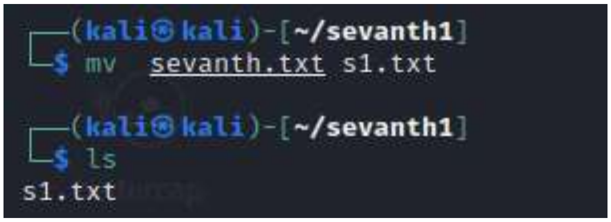
4. **List all the files and directories in the current directory.**



```
┌──(kali㉿kali)-[~/sevanth1]
└─$ ls
sevanth.txt
```

The **ls** command lists the files and directories in the current directory.

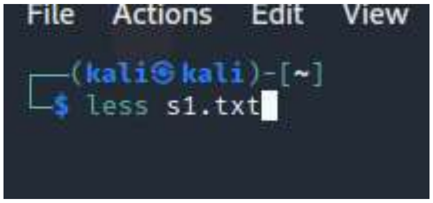
5. **Rename "my_file.txt" to "new_file.txt".**



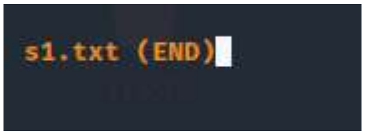
```
┌──(kali㉿kali)-[~/sevanth1]
└─$ mv  sevanth.txt s1.txt

┌──(kali㉿kali)-[~/sevanth1]
└─$ ls
s1.txt
```

The **mv** command is used to move or rename files. In this case, it renames the file "my_file.txt" to "new_file.txt"

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



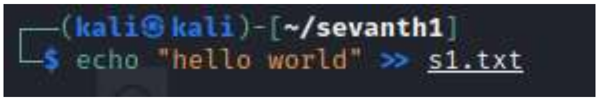
```
┌──(kali㉿kali)-[~]
└─$ less s1.txt
```



```
s1.txt (END)
```

The **less** command is a pager tool that allows you to view the content of a file page by page. In this case, it displays the content of the file "new_file.txt". You can scroll through the content using the arrow keys and press "q" to exit.

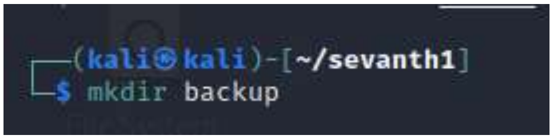
7. **Append the text "Hello, World!" to "new_file.txt".**



```
┌──(kali㉿kali)-[~/sevanth1]
└─$ echo "hello world" >> s1.txt
```

The **echo** command is used to print text. The >> operator is used to append the output to a file. In this case, it appends the text "Hello, World!" to the file "new_file.txt"

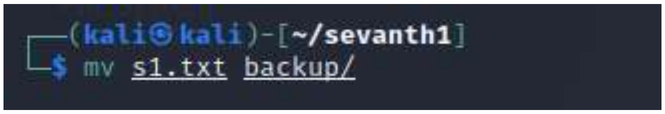
8. **Create a new directory called "backup" within "my_directory".**



```
┌──(kali㉿kali)-[~/sevanth1]
└─$ mkdir backup
```

This command creates a new directory named "backup" within the "my_directory" directory.

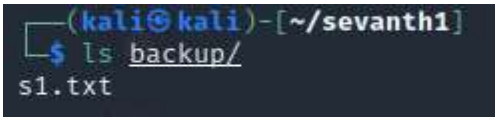
9. **Move "new_file.txt" to the "backup" directory.**



```
┌──(kali㉿kali)-[~/sevanth1]
└─$ mv s1.txt backup/
```

This command moves the file "new_file.txt" to the "backup" directory.

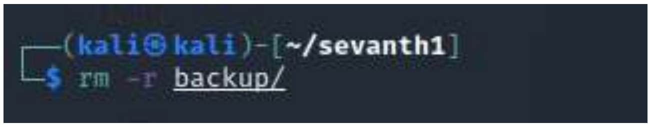
10. **Verify that "new_file.txt" is now located in the "backup" directory.**



```
┌──(kali㉿kali)-[~/sevanth1]
└─$ ls backup/
s1.txt
```

This command lists the contents of the "backup" directory to verify that "new_file.txt" is present there.

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

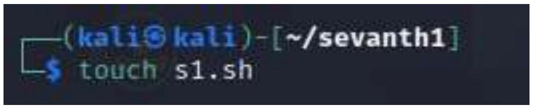


```
┌──(kali㉿kali)-[~/sevanth1]
└─$ rm -r backup/
```

The **rm** command is used to remove files and directories. The **-r** option is used to recursively remove directories and their contents. In this case, it deletes the "backup" directory and all its contents

Task 2: Permissions and Scripting

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



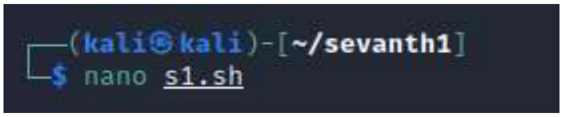
```
┌──(kali㉿kali)-[~/sevanth1]
└─$ touch s1.sh
```

This command creates a new file named "my_script.sh" in the current directory.

- **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.

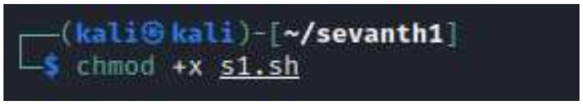


```
┌──(kali㉿kali)-[~/sevanth1]
└─$ nano s1.sh
```

This command opens the "my_script.sh" file in the nano text editor, allowing you to edit the file

These lines are added to the "my_script.sh" file. The first line specifies the interpreter (**#!/bin/bash**), and the subsequent lines use the **echo** command to print text.

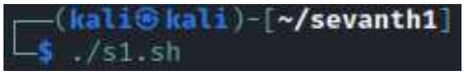
- **Make "my_script.sh" executable**



```
┌──(kali㉿kali)-[~/sevanth1]
└─$ chmod +x s1.sh
```

The **chmod** command is used to change the permissions of a file. The **+x** option makes the file executable, allowing it to be run as a script.

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

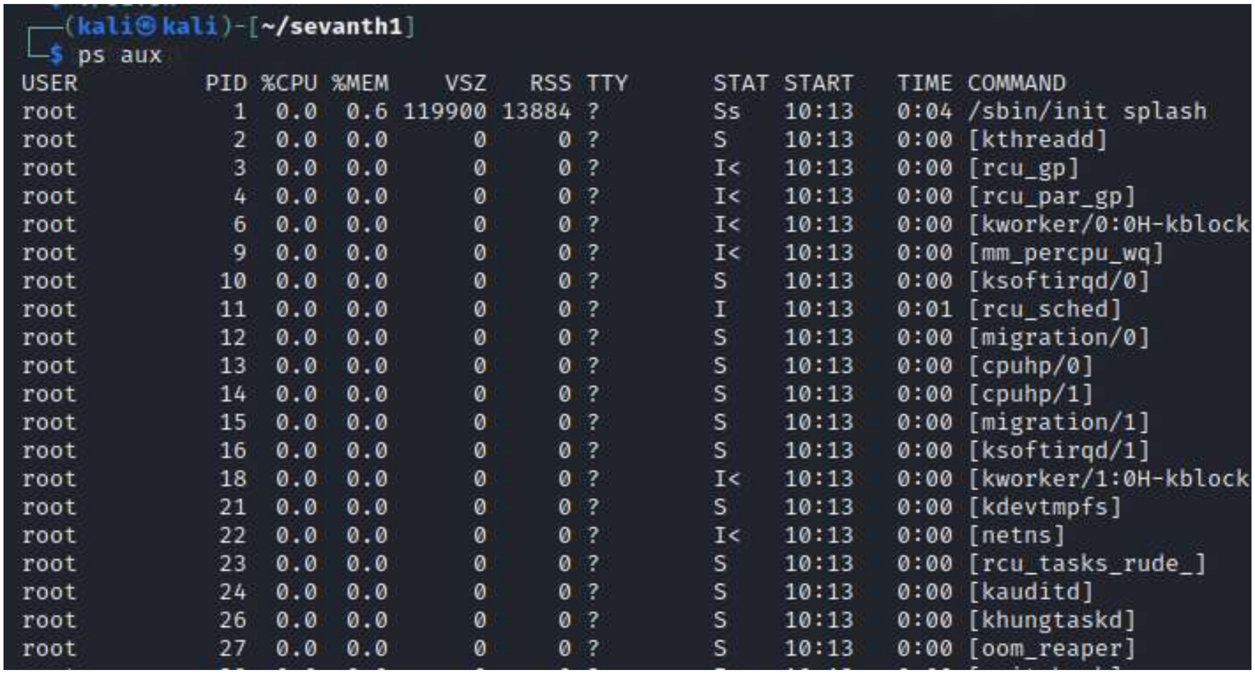


```
┌──(kali㉿kali)-[~/sevanth1]
└─$ ./s1.sh
```

This command executes the "my_script.sh" file, and the output should display the text specified in the script, including the current date and time

Task 3: Command Execution and Pipelines

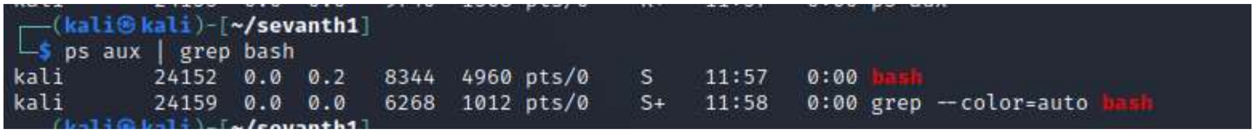
- **List all the processes running on your system using the "ps" command.**



```
┌──(kali㉿kali)-[~/sevanth1]
└─$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.6 119900 13884 ?        Ss   10:13   0:04 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    10:13   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   10:13   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   10:13   0:00 [rcu_par_gp]
root           6  0.0  0.0      0     0 ?        I<   10:13   0:00 [kworker/0:0H-kblock
root           9  0.0  0.0      0     0 ?        I<   10:13   0:00 [mm_percpu_wq]
root          10  0.0  0.0      0     0 ?        S    10:13   0:00 [ksoftirqd/0]
root          11  0.0  0.0      0     0 ?        I    10:13   0:01 [rcu_sched]
root          12  0.0  0.0      0     0 ?        S    10:13   0:00 [migration/0]
root          13  0.0  0.0      0     0 ?        S    10:13   0:00 [cpuhp/0]
root          14  0.0  0.0      0     0 ?        S    10:13   0:00 [cpuhp/1]
root          15  0.0  0.0      0     0 ?        S    10:13   0:00 [migration/1]
root          16  0.0  0.0      0     0 ?        S    10:13   0:00 [ksoftirqd/1]
root          18  0.0  0.0      0     0 ?        I<   10:13   0:00 [kworker/1:0H-kblock
root          21  0.0  0.0      0     0 ?        S    10:13   0:00 [kdevtmpfs]
root          22  0.0  0.0      0     0 ?        I<   10:13   0:00 [netns]
root          23  0.0  0.0      0     0 ?        S    10:13   0:00 [rcu_tasks_rude_]
root          24  0.0  0.0      0     0 ?        S    10:13   0:00 [kauditd]
root          26  0.0  0.0      0     0 ?        S    10:13   0:00 [khungtaskd]
root          27  0.0  0.0      0     0 ?        S    10:13   0:00 [oom_reaper]
```

The **ps** command is used to display information about active processes. The **aux** options provide a detailed list of all processes running on the system.

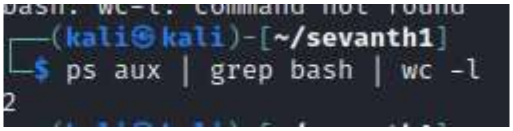
- **Use the "grep" command to filter the processes list and display only the processes with "bash" in their name.**



```
┌──(kali㉿kali)-[~/sevanth1]
└─$ ps aux | grep bash
kali       24152  0.0  0.2   8344  4960 pts/0    S    11:57   0:00 bash
kali       24159  0.0  0.0   6268  1012 pts/0    S+   11:58   0:00 grep --color=auto bash
```

The **grep** command is used to search for specific patterns in the input. In this case, it filters the output of the **ps aux** command to display only the processes that contain the word "bash"

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



```
bash: wc-l: command not found
┌──(kali㉿kali)-[~/sevanth1]
└─$ ps aux | grep bash | wc -l
2
```

The **wc** command is used to count the number of lines, words, and characters in the input. The **-l** option tells **wc** to count only the lines. In this case, it counts the number of lines in the filtered output of the previous command, giving the total number of processes with "bash" in their name.