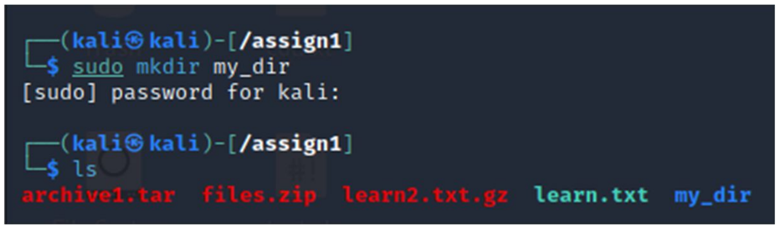
NILADRI MITRA
VIT REG.NO : 20BIT0381
COURSE : CYBERSECURITY & ETHICAL HACKING(SMARTBRIDGE EXTERNSHIP)
Vit email id : niladri.mitra2020@vitstudent.ac.in

ASSESSMENT – 2

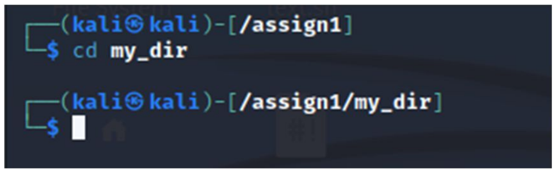
TASK1 . File and directory manipulation

1. Create a directory called "my_directory".
mkdir my_directory
ls



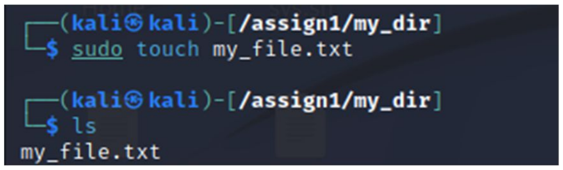
```
┌──(kali㉿kali)-[/assign1]
└─$ sudo mkdir my_dir
[sudo] password for kali:

┌──(kali㉿kali)-[/assign1]
└─$ ls
archive1.tar  files.zip  learn2.txt.gz  learn.txt  my_dir
```

2. Navigate into the "my_directory".
cd my_directory



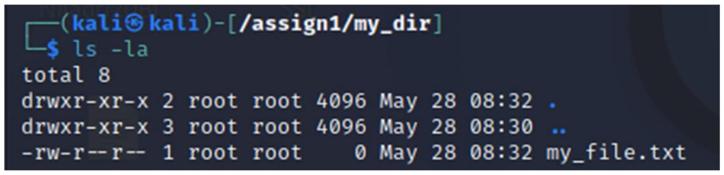
```
┌──(kali㉿kali)-[/assign1]
└─$ cd my_dir

┌──(kali㉿kali)-[/assign1/my_dir]
└─$ 
```

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



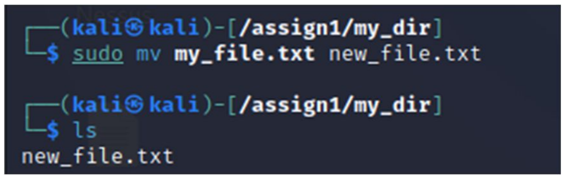
```
┌──(kali㉿kali)-[/assign1/my_dir]
└─$ sudo touch my_file.txt

┌──(kali㉿kali)-[/assign1/my_dir]
└─$ ls
my_file.txt
```

4. List all the files and directories in the current directory.
ls -la



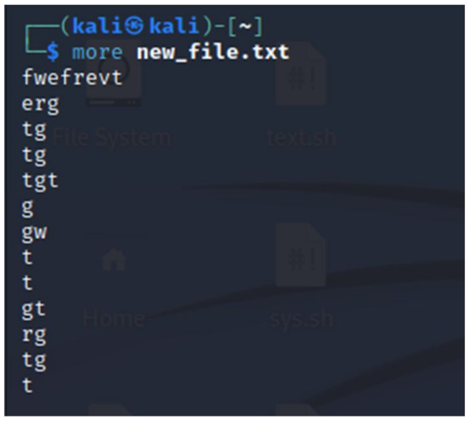
```
┌──(kali㉿kali)-[/assign1/my_dir]
└─$ ls -la
total 8
drwxr-xr-x 2 root root 4096 May 28 08:32 .
drwxr-xr-x 3 root root 4096 May 28 08:30 ..
-rw-r--r-- 1 root root    0 May 28 08:32 my_file.txt
```

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



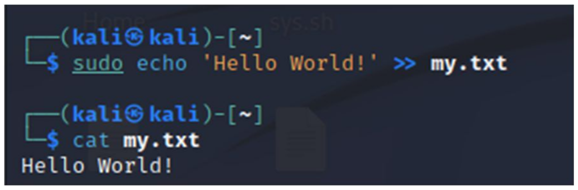
```
┌──(kali㉿kali)-[/assign1/my_dir]
└─$ sudo mv my_file.txt new_file.txt

┌──(kali㉿kali)-[/assign1/my_dir]
└─$ ls
new_file.txt
```

6. Display the content of "new_file.txt" using a pager tool of your choice.
more new_file.txt
(to display content of new_file.txt , I have added random words to it)



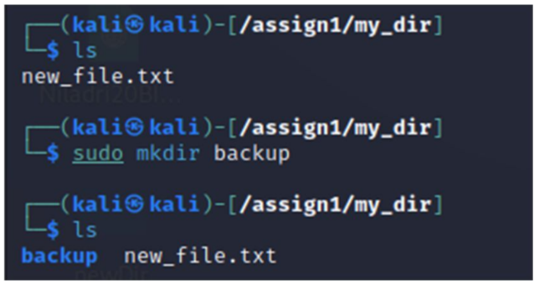
```
┌──(kali㉿kali)-[~]
└─$ more new_file.txt
fwefrevt
erg
tg
tg
tgt
g
gw
t
t
gt
rg
tg
t
```

7. Append the text "Hello, World!" to "new_file.txt".
echo ‘Hello, World!’ >> new_file.txt



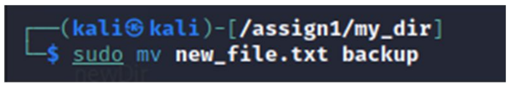
```
┌──(kali㉿kali)-[~]
└─$ sudo echo 'Hello World!' >> my.txt

┌──(kali㉿kali)-[~]
└─$ cat my.txt
Hello World!
```

8. Create a new directory called "backup" within "my_directory".
ls
mkdir backup
ls



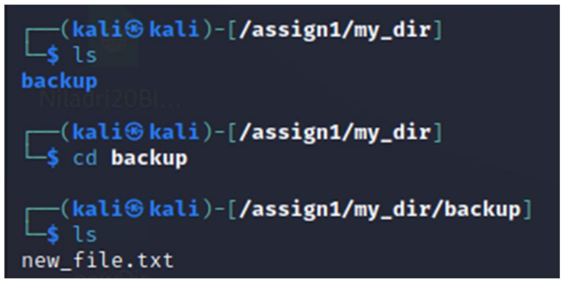
```
┌──(kali㉿kali)-[/assign1/my_dir]
└─$ ls
new_file.txt

┌──(kali㉿kali)-[/assign1/my_dir]
└─$ sudo mkdir backup

┌──(kali㉿kali)-[/assign1/my_dir]
└─$ ls
backup  new_file.txt
```

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



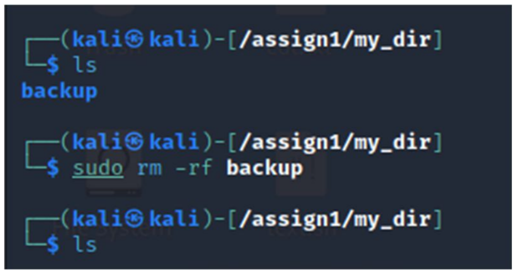
```
┌──(kali㉿kali)-[/assign1/my_dir]
└─$ sudo mv new_file.txt backup
```

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



```
┌──(kali㉿kali)-[/assign1/my_dir]
└─$ ls
backup

┌──(kali㉿kali)-[/assign1/my_dir]
└─$ cd backup

┌──(kali㉿kali)-[/assign1/my_dir/backup]
└─$ ls
new_file.txt
```

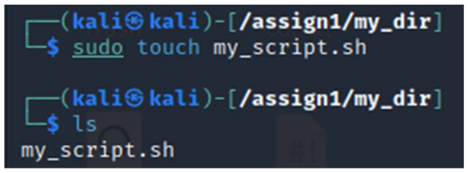
11. Delete the "backup" directory and all its contents.
ls
rm -rf backup
ls



```
┌──(kali㉿kali)-[/assign1/my_dir]
└─$ ls
backup

┌──(kali㉿kali)-[/assign1/my_dir]
└─$ sudo rm -rf backup

┌──(kali㉿kali)-[/assign1/my_dir]
└─$ ls
```

TASK 2 : PERMISSIONS AND SCRIPTING

1.Create a new file called “my_script.sh”
touch my_script.sh
ls

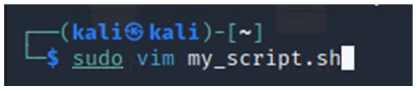


```
┌──(kali㉿kali)-[/assign1/my_dir]
└─$ sudo touch my_script.sh

┌──(kali㉿kali)-[/assign1/my_dir]
└─$ ls
my_script.sh
```

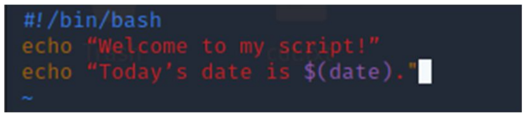
2. Edit my_script.sh using any text editor , add the given lines, make it executable , and run.
vim my_script.sh
#!/bin/bash
echo “Welcome to my script!”
echo “Today’s date is $(date).”

w : save changes made to the file
q : exit Vim

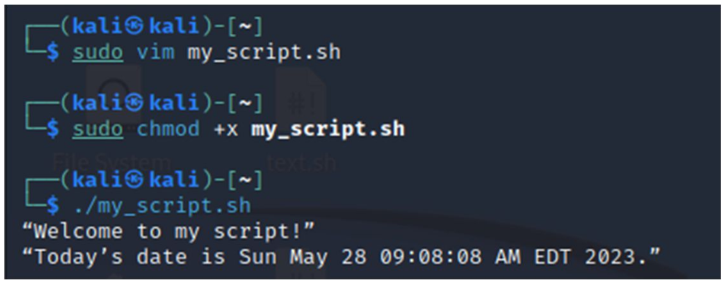
chmod +x my_script.sh
./my_script.sh



```
┌──(kali㉿kali)-[~]
└─$ sudo vim my_script.sh
```



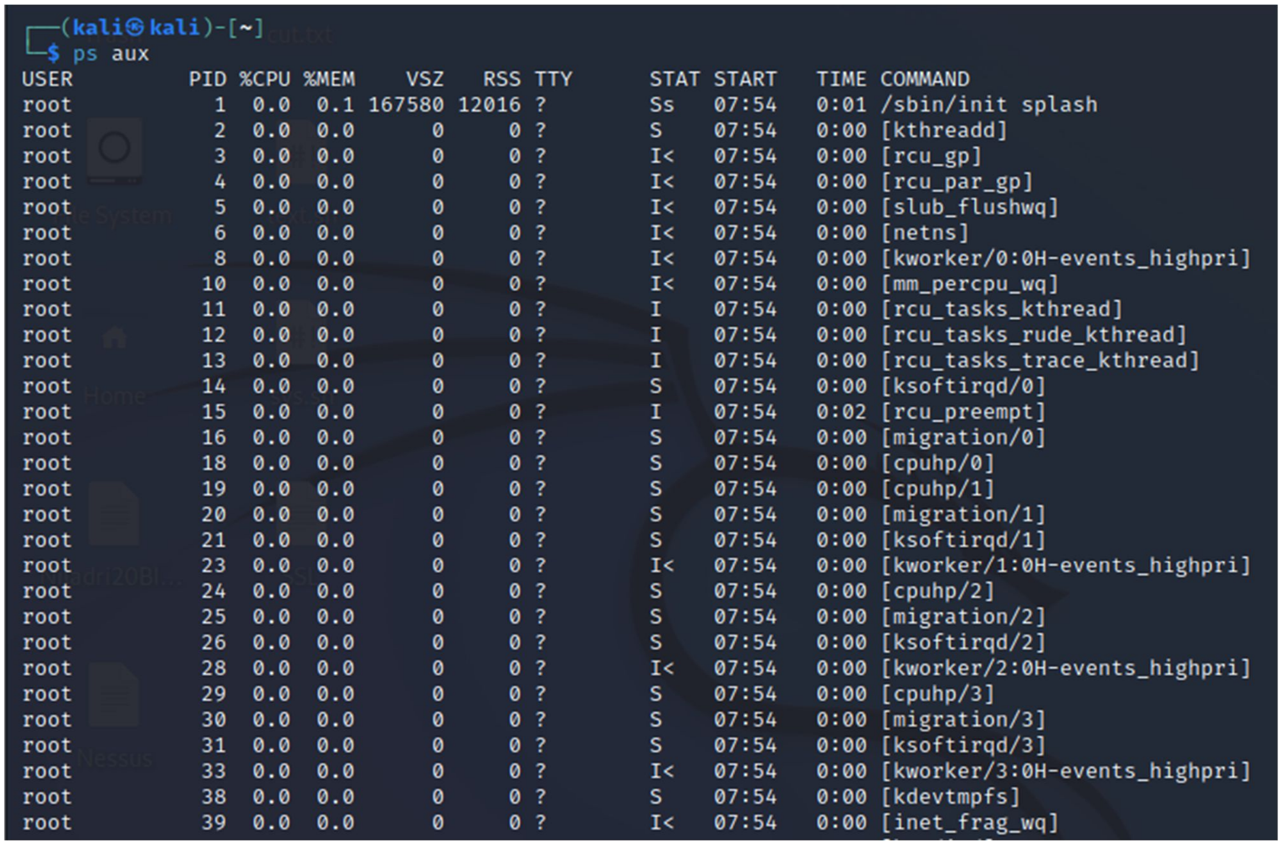
```
#!/bin/bash
echo “Welcome to my script!”
echo “Today’s date is $(date).”
~
```



```
┌──(kali㉿kali)-[~]
└─$ sudo vim my_script.sh

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

┌──(kali㉿kali)-[~]
└─$ ./my_script.sh
“Welcome to my script!”
“Today’s date is Sun May 28 09:08:08 AM EDT 2023.”
```

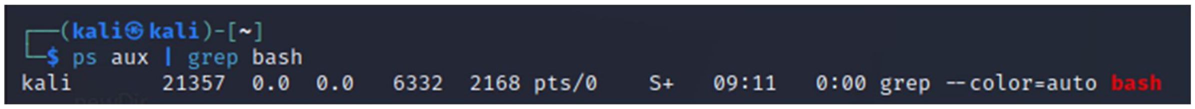
TASK 3 : COMMAND EXECUTION AND PIPELINES

1. List all the processes running on your system using the "ps" command.
ps aux
The ps aux command is used to display a detailed list of all running processes on a Linux or Unix system.



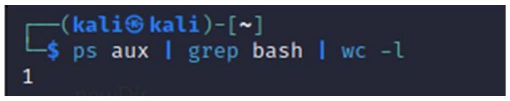
```
┌──(kali㉿kali)-[~]
└─$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.1 167580 12016 ?        Ss   07:54   0:01 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    07:54   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   07:54   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   07:54   0:00 [rcu_par_gp]
root           5  0.0  0.0      0     0 ?        I<   07:54   0:00 [slub_flushwq]
root           6  0.0  0.0      0     0 ?        I<   07:54   0:00 [netns]
root           8  0.0  0.0      0     0 ?        I<   07:54   0:00 [kworker/0:0H-events_highpri]
root          10  0.0  0.0      0     0 ?        I<   07:54   0:00 [mm_percpu_wq]
root          11  0.0  0.0      0     0 ?        I    07:54   0:00 [rcu_tasks_kthread]
root          12  0.0  0.0      0     0 ?        I    07:54   0:00 [rcu_tasks_rude_kthread]
root          13  0.0  0.0      0     0 ?        I    07:54   0:00 [rcu_tasks_trace_kthread]
root          14  0.0  0.0      0     0 ?        S    07:54   0:00 [ksoftirqd/0]
root          15  0.0  0.0      0     0 ?        I    07:54   0:02 [rcu_preempt]
root          16  0.0  0.0      0     0 ?        S    07:54   0:00 [migration/0]
root          18  0.0  0.0      0     0 ?        S    07:54   0:00 [cpuhp/0]
root          19  0.0  0.0      0     0 ?        S    07:54   0:00 [cpuhp/1]
root          20  0.0  0.0      0     0 ?        S    07:54   0:00 [migration/1]
root          21  0.0  0.0      0     0 ?        S    07:54   0:00 [ksoftirqd/1]
root          23  0.0  0.0      0     0 ?        I<   07:54   0:00 [kworker/1:0H-events_highpri]
root          24  0.0  0.0      0     0 ?        S    07:54   0:00 [cpuhp/2]
root          25  0.0  0.0      0     0 ?        S    07:54   0:00 [migration/2]
root          26  0.0  0.0      0     0 ?        S    07:54   0:00 [ksoftirqd/2]
root          28  0.0  0.0      0     0 ?        I<   07:54   0:00 [kworker/2:0H-events_highpri]
root          29  0.0  0.0      0     0 ?        S    07:54   0:00 [cpuhp/3]
root          30  0.0  0.0      0     0 ?        S    07:54   0:00 [migration/3]
root          31  0.0  0.0      0     0 ?        S    07:54   0:00 [ksoftirqd/3]
root          33  0.0  0.0      0     0 ?        I<   07:54   0:00 [kworker/3:0H-events_highpri]
root          38  0.0  0.0      0     0 ?        S    07:54   0:00 [kdevtmpfs]
root          39  0.0  0.0      0     0 ?        I<   07:54   0:00 [inet_frag_wq]
```

2.Use the "grep" command to filter the processes list and display only the processes with "bash" in their name.
ps aux | grep bash
(grep is used for matching a pattern or string)



```
┌──(kali㉿kali)-[~]
└─$ ps aux | grep bash
kali       21357  0.0  0.0   6332  2168 pts/0    S+   09:11   0:00 grep --color=auto bash
```

3.Use the "wc" command to count the number of lines in the filtered output.
ps aux | grep bash | wc -l



```
┌──(kali㉿kali)-[~]
└─$ ps aux | grep bash | wc -l
1
```