CYBERSECURITY & ETHICAL HACKING
BASH SHELL BASICS
ASSIGNMENT - II

KHUSHI JAIN
20BCE2664

I. FILE AND DIRECTORY MANIPULATION

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

2. Navigate into the "my_directory".
   → cd my_directory

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

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

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

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

7. Append the text "Hello, World!" to "new_file.txt".
   → echo “Hello, World!” >> new_file.txt

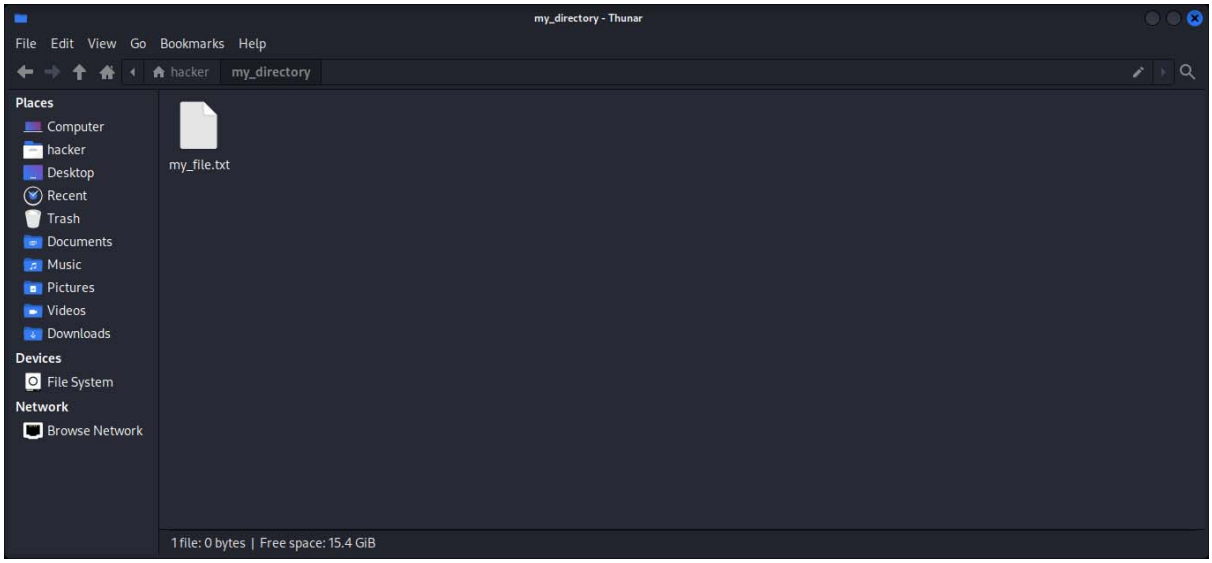
8. Create a new directory called "backup" within "my_directory".
   → mkdir backup

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

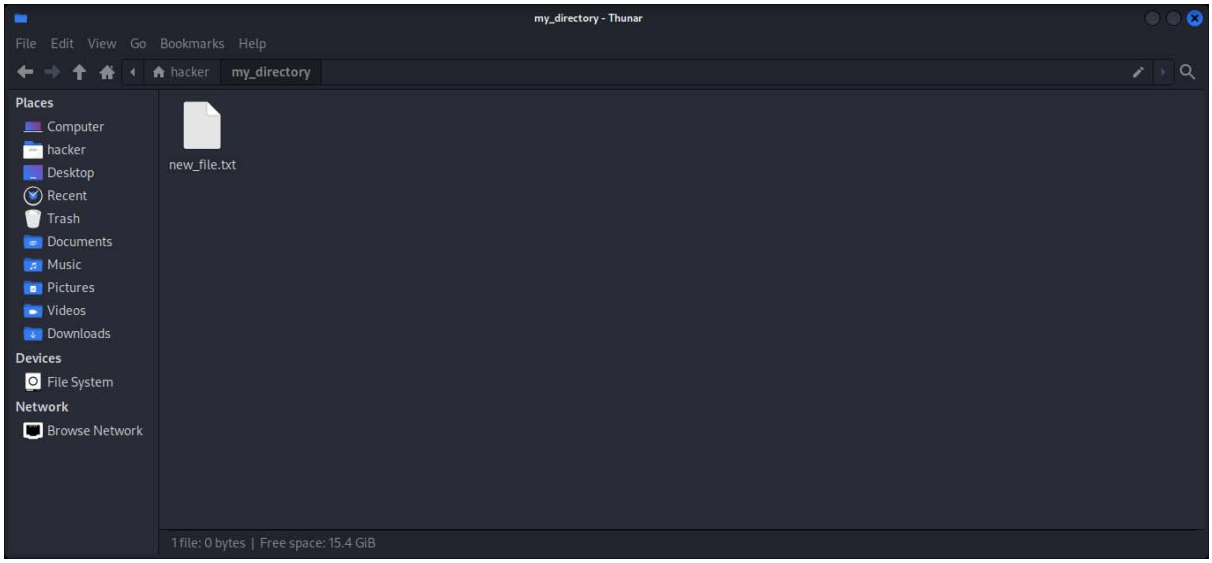
10. Verify that "new_file.txt" is now located in the "backup" directory.
    → ls backup

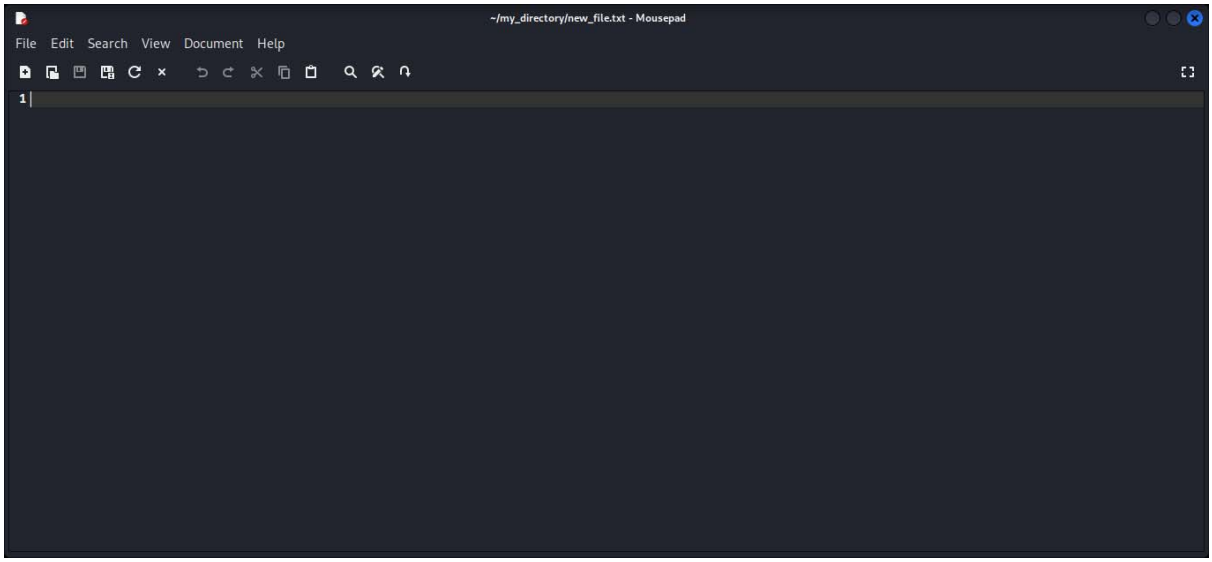
11. Delete the "backup" directory and all its contents.
    → rm –rf backup

Output of Commands 1 - 4

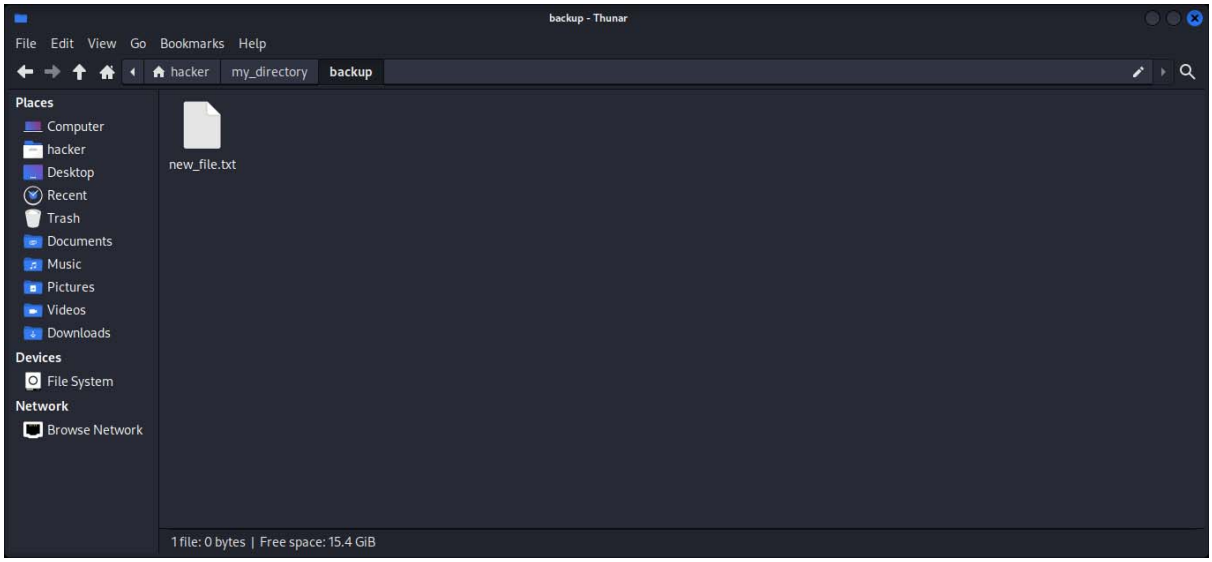


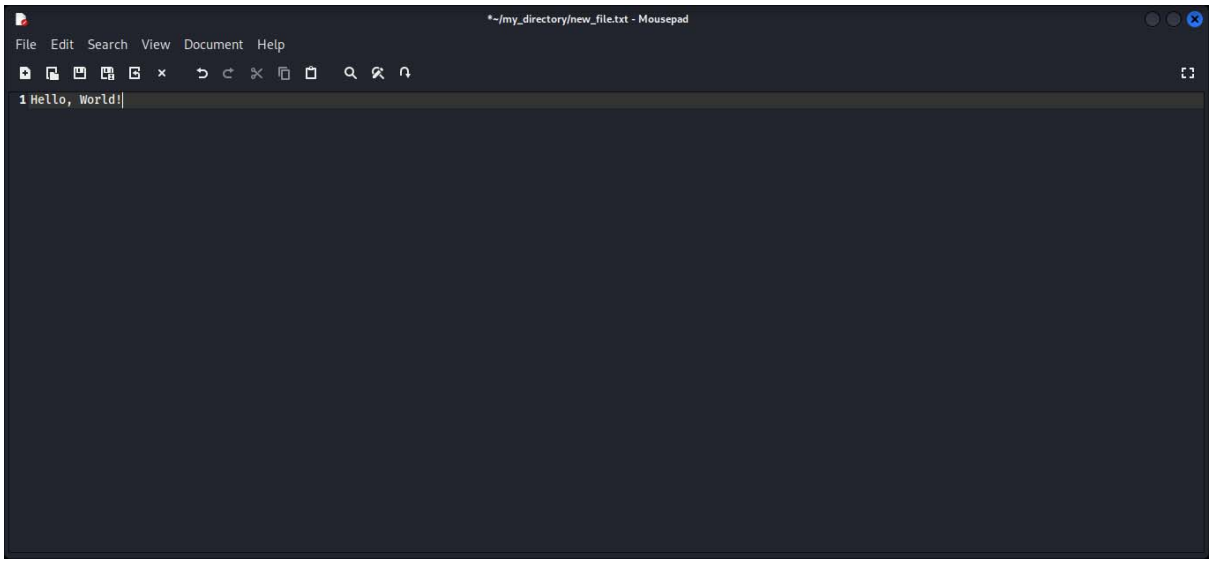
Output of Commands 1 - 6





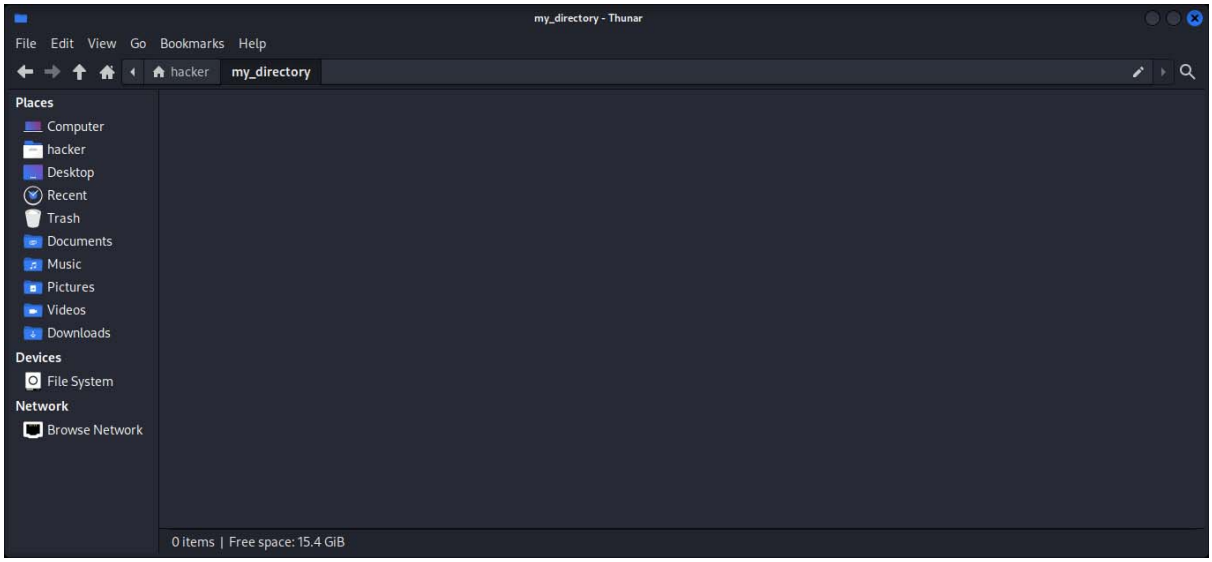
Output of Commands 1 - 10


backup - Thunar
File Edit View Go Bookmarks Help
hacker
my_directory
backup
Places
Computer
hacker
Desktop
Recent
Trash
Documents
Music
Pictures
Videos
Downloads
Devices
File System
Network
Browse Network
new_file.txt
1 file: 0 bytes | Free space: 15.4 GiB


*~/my_directory/new_file.txt - Mousepad
File Edit Search View Document Help
1 Hello, World!


Shell No. 1
File Actions Edit View Help
#!/bin/bash
#1. Create a directory called "my_directory".
mkdir my_directory
#2. Navigate into the "my_directory".
cd my_directory
#3. Create an empty file called "my_file.txt".
touch my_file.txt
#4. List all the files and directories in the current directory.
ls
#5. Rename "my_file.txt" to "new_file.txt"
mv my_file.txt new_file.txt
#6. Display the content of "new_file.txt" using a pager tool of you choice.
cat new_file.txt | less
#7. Append the text "Hello, World!" to "new_file.txt".
echo "Hello, World!" >> new_file.txt
#8. Create a new directory called "backup" within "my_directory".
mkdir backup
#9. Move "new_file.txt" to the "backup" directory.
mv new_file.txt backup
#10. Verify that "new_file.txt" is now located in the "backup" directory.
ls backup
#11. Delete the "backup" directory and all its contents.
28,22 Top

Output of Commands 1 - 11



Explanation

The code (1- 4) first creates a directory called "my_directory". Then, it navigates into the "my_directory" directory. Next, it creates an empty file called "my_file.txt". Then, it lists all the files and directories in the current directory.

The code (1- 6) then renames "my_file.txt" to "new_file.txt". It then displays the content of "new_file.txt" using a pager tool of your choice. Next, it appends the text "Hello, World!" to "new_file.txt".

The code (1- 10) then creates a new directory called "backup" within "my_directory". It then moves "new_file.txt" to the "backup" directory. Finally, it verifies that "new_file.txt" is now located in the "backup" directory.

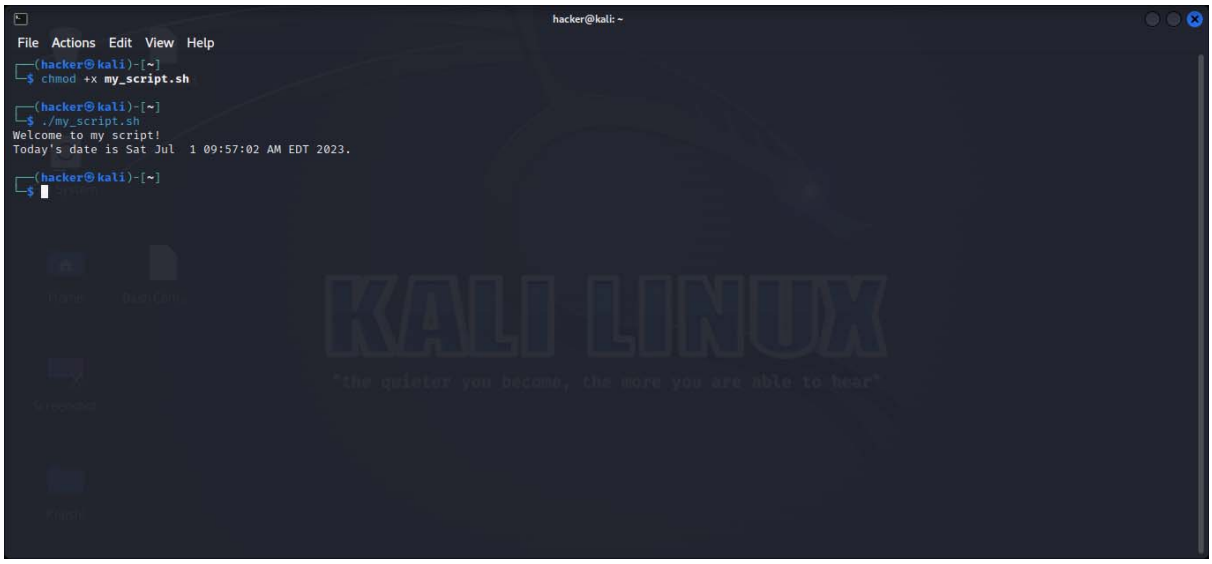
The code (1- 11) then deletes the "backup" directory and all its contents.

II. 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.



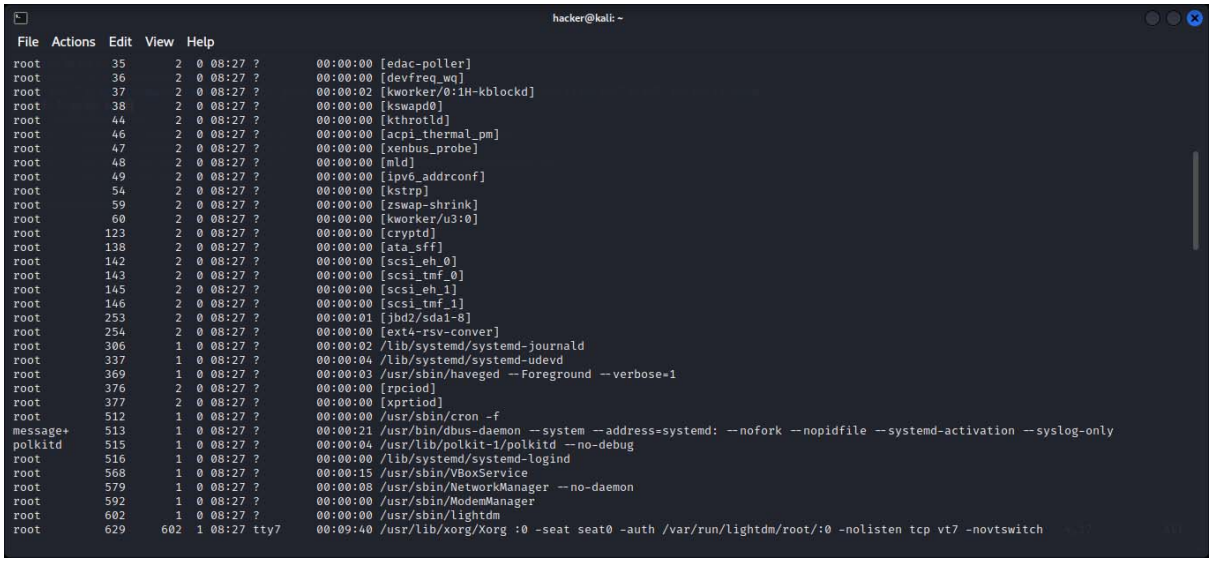
Explanation

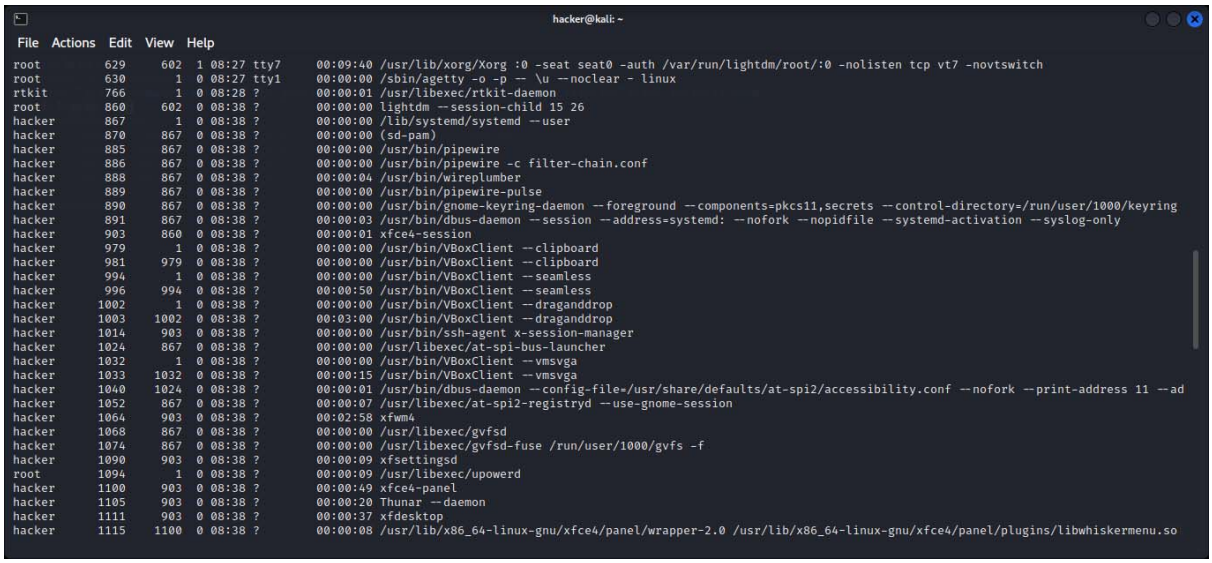
The first line, #!/bin/bash, tells the operating system that the file is a Bash script and that it should be executed using the Bash shell. The second line, echo “Welcome to my script!” will print the text "Welcome to my script!" to the console. The third line, echo “Today's date is $(date).” will print the current date to the console.

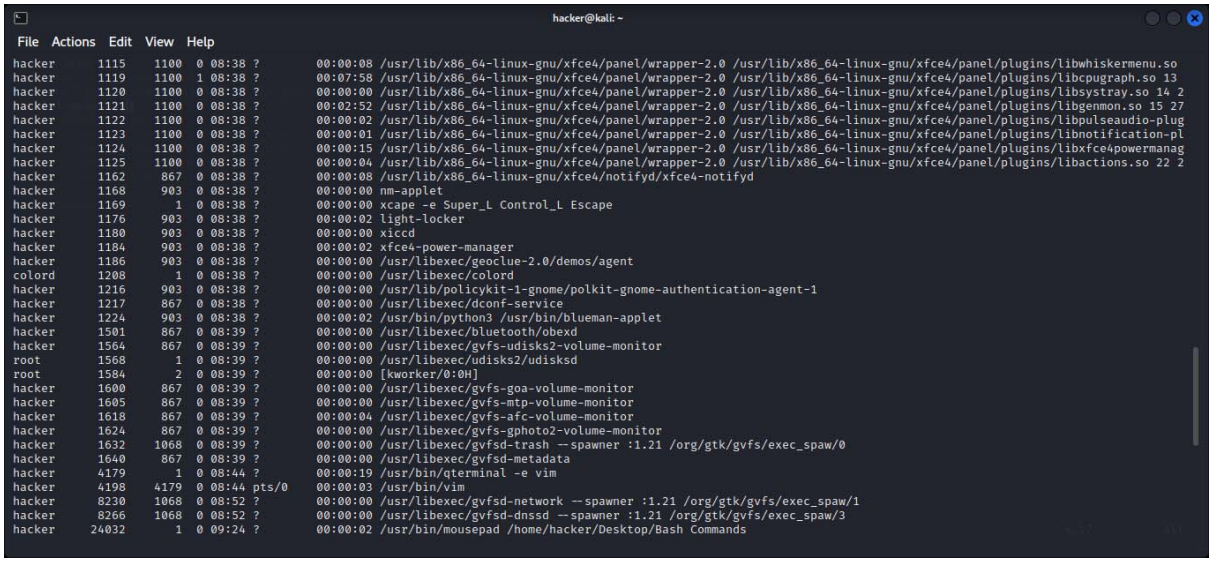
III. COMMAND EXECUTION AND PIPELINES

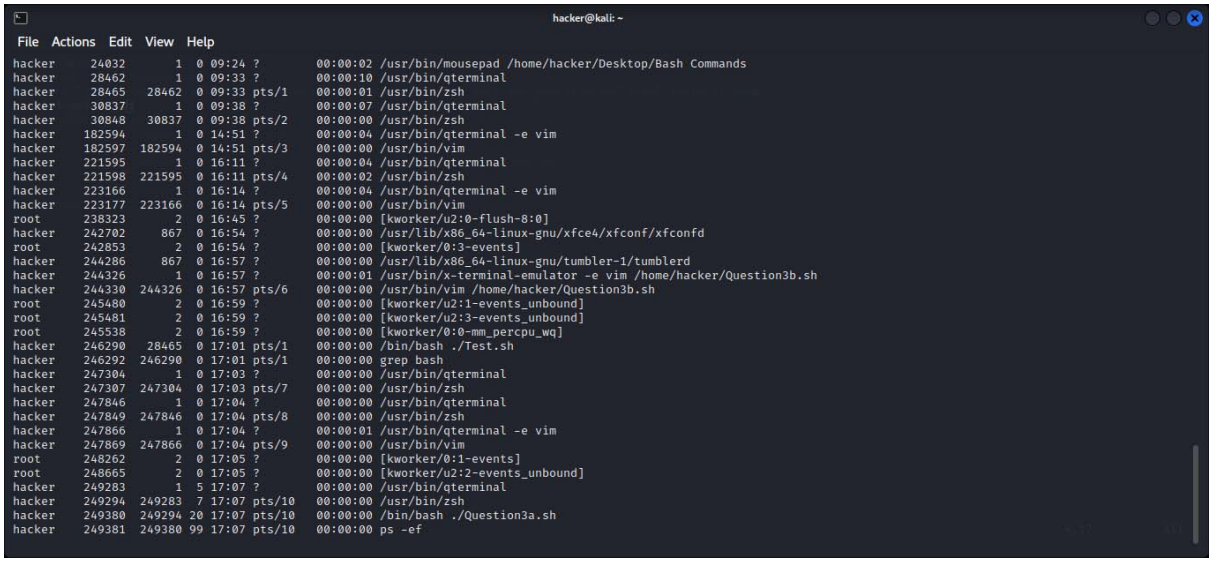
- List all the processes running on your system using the "ps" command.

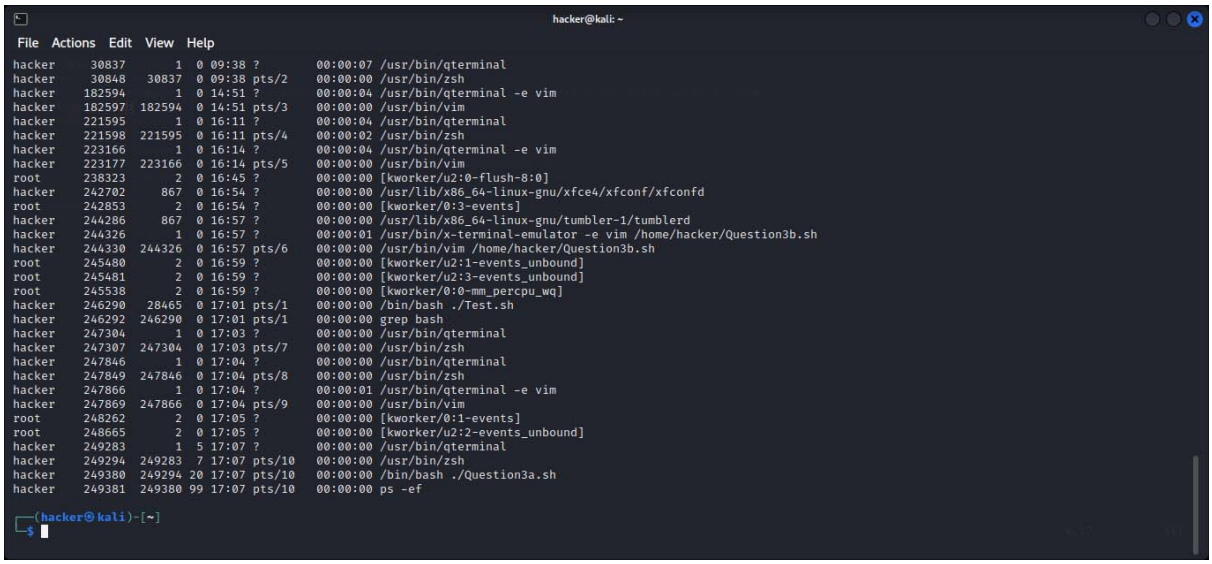
  → ps –ef





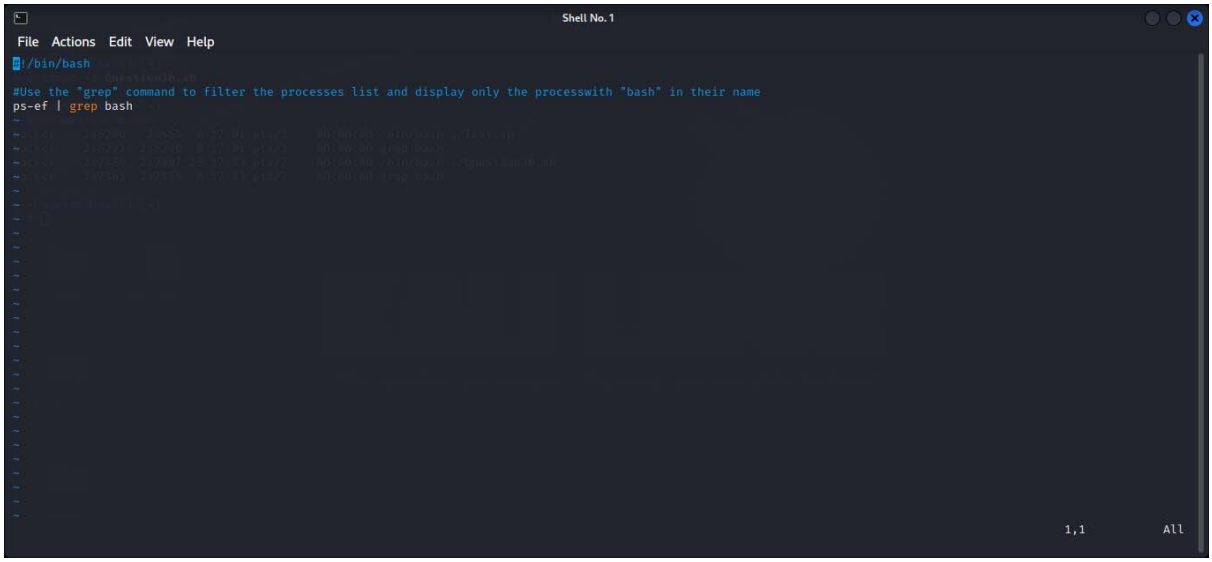




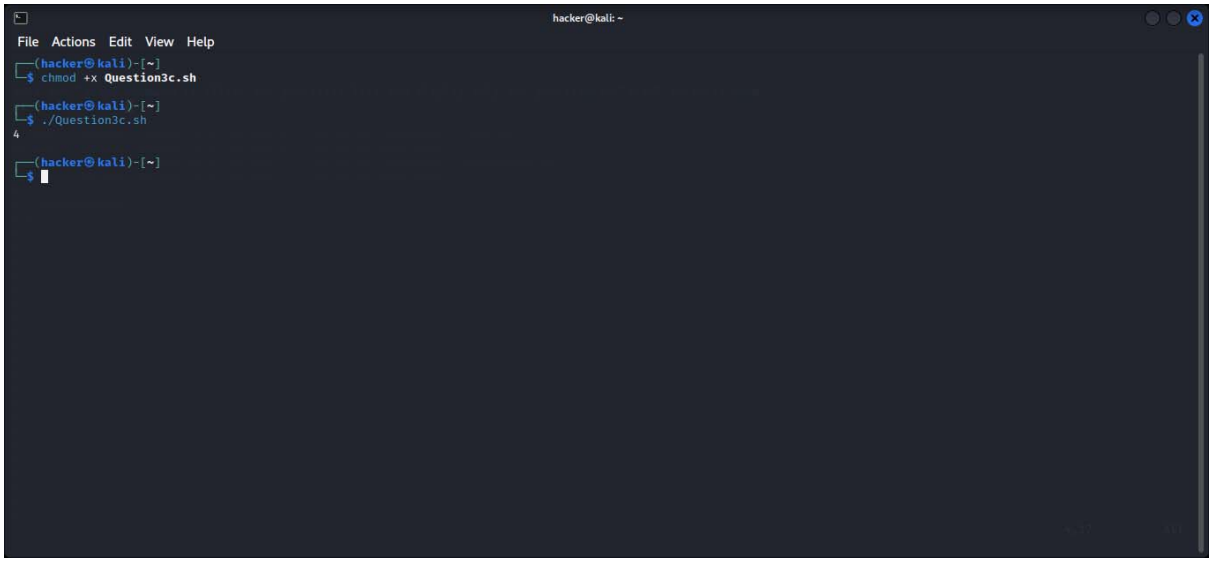


- Use the "grep" command to filter the processes list and display only the processes with "bash" in their name.

  → ps –ef | grep bash



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



Explanation

The first line, #!/bin/bash, tells the operating system that the file is a Bash script and that it should be executed using the Bash shell.

The second line, ps -ef, lists all the processes running on the system. This command will output a list of all the processes, including their name, PID, user, and command line.

The third line, ps -ef | grep bash, filters the processes list and displays only the processes with "bash" in their name. The grep command is a text search utility that can be used to search for a specific pattern in a file or stream of data. In this case, the grep command is searching for the string "bash" in the output of the ps -ef command.

The fourth line, ps -ef | grep bash | wc -l, counts the number of lines in the filtered output. The wc command is a word count utility that can be used to count the number of lines, words, or characters in a file or stream of data. In this case, the wc command is counting the number of lines in the output of the grep command.