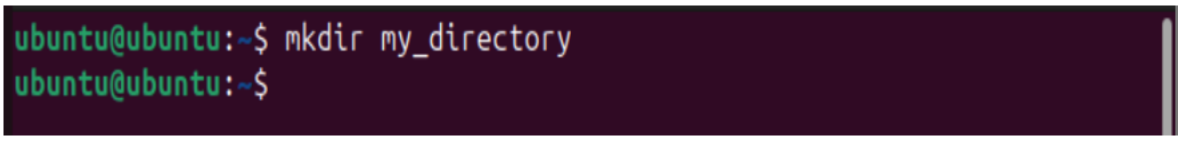
Name: P. Nithiyasri Reg No.: 20BCE7035

ASSIGNMENT 2

BASH SHELL BASICS

TASK 1: File and Directory Manipulation

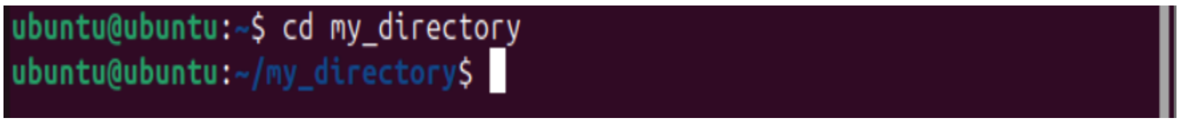
1. Create a directory called "my_directory".



```
ubuntu@ubuntu:~$ mkdir my_directory
ubuntu@ubuntu:~$
```

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

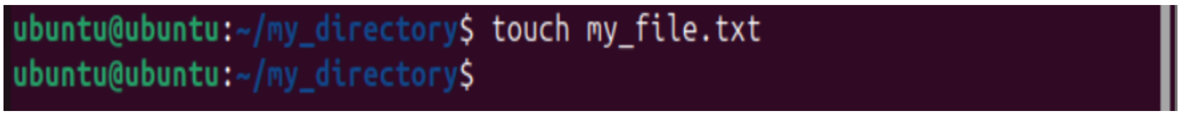
2. Navigate into the "my_directory".



```
ubuntu@ubuntu:~$ cd my_directory
ubuntu@ubuntu:~/my_directory$
```

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

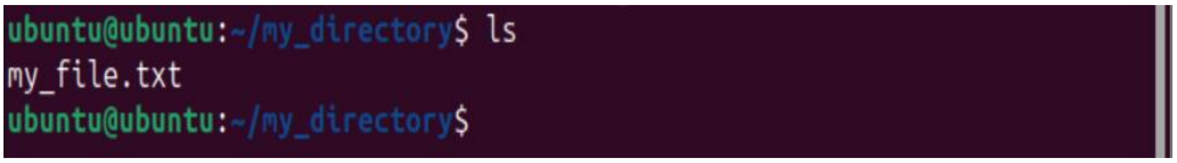
3. Create an empty file called "my_file.txt".



```
ubuntu@ubuntu:~/my_directory$ touch my_file.txt
ubuntu@ubuntu:~/my_directory$
```

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.**

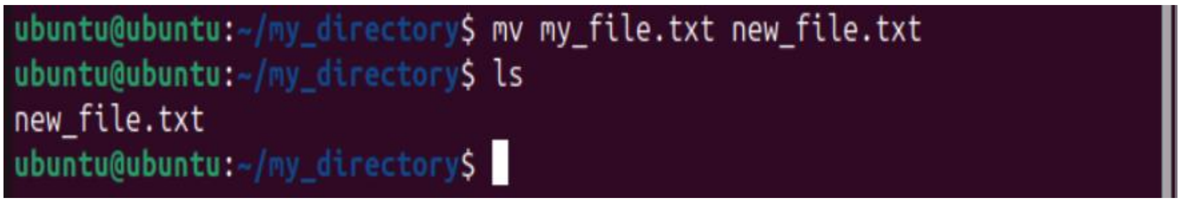
4. List all the files and directories in the current directory.



```
ubuntu@ubuntu:~/my_directory$ ls
my_file.txt
ubuntu@ubuntu:~/my_directory$
```

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

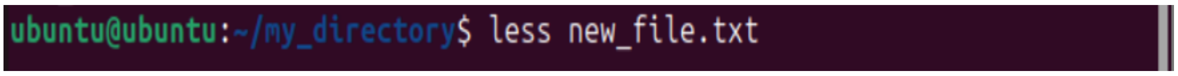
5. Rename "my_file.txt" to "new_file.txt".



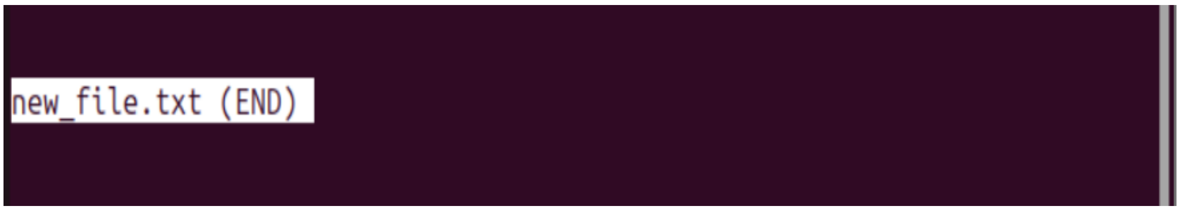
```
ubuntu@ubuntu:~/my_directory$ mv my_file.txt new_file.txt
ubuntu@ubuntu:~/my_directory$ ls
new_file.txt
ubuntu@ubuntu:~/my_directory$
```

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



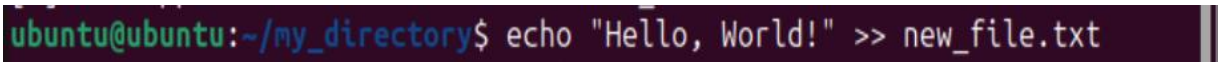
```
ubuntu@ubuntu:~/my_directory$ less new_file.txt
```



```
new_file.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.**

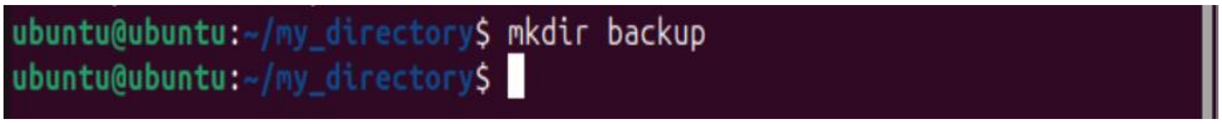
7. Append the text "Hello, World!" to "new_file.txt".



```
ubuntu@ubuntu:~/my_directory$ echo "Hello, World!" >> new_file.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".**

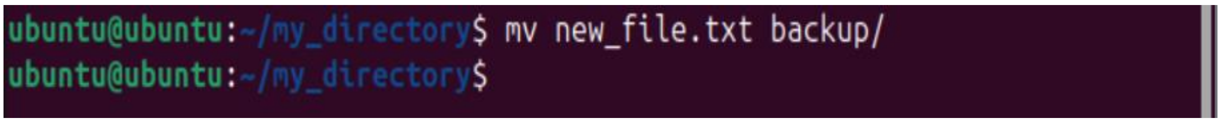
8. Create a new directory called "backup" within "my_directory".



```
ubuntu@ubuntu:~/my_directory$ mkdir backup
ubuntu@ubuntu:~/my_directory$ 
```

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

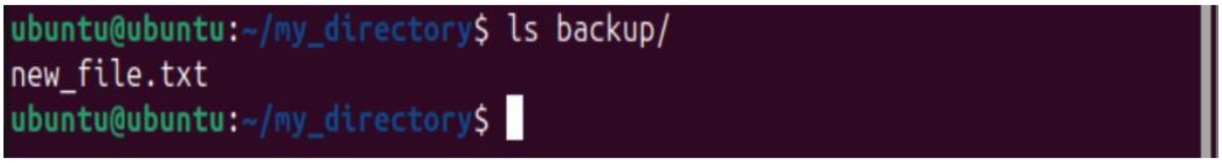
9. Move "new_file.txt" to the "backup" directory.



```
ubuntu@ubuntu:~/my_directory$ mv new_file.txt backup/
ubuntu@ubuntu:~/my_directory$
```

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

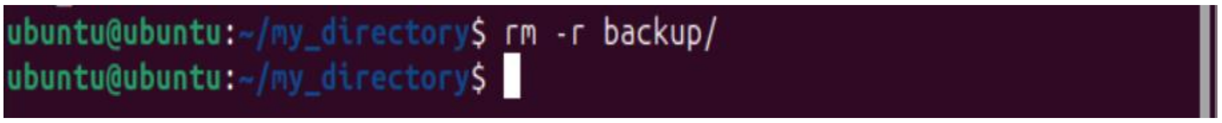
10. Verify that "new_file.txt" is now located in the "backup" directory.



```
ubuntu@ubuntu:~/my_directory$ ls backup/
new_file.txt
ubuntu@ubuntu:~/my_directory$ 
```

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.

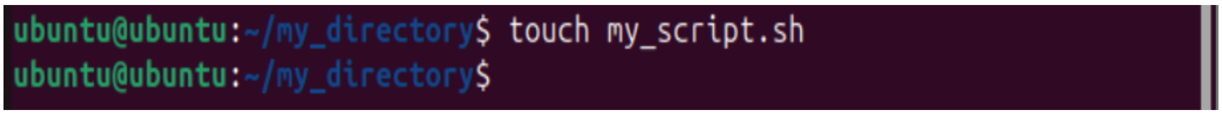


```
ubuntu@ubuntu:~/my_directory$ rm -r backup/
ubuntu@ubuntu:~/my_directory$ 
```

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".



```
ubuntu@ubuntu:~/my_directory$ touch my_script.sh
ubuntu@ubuntu:~/my_directory$
```

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:

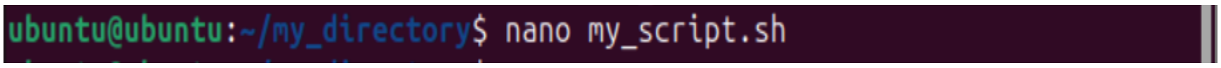
bash

#!/bin/bash

echo "Welcome to my script!"

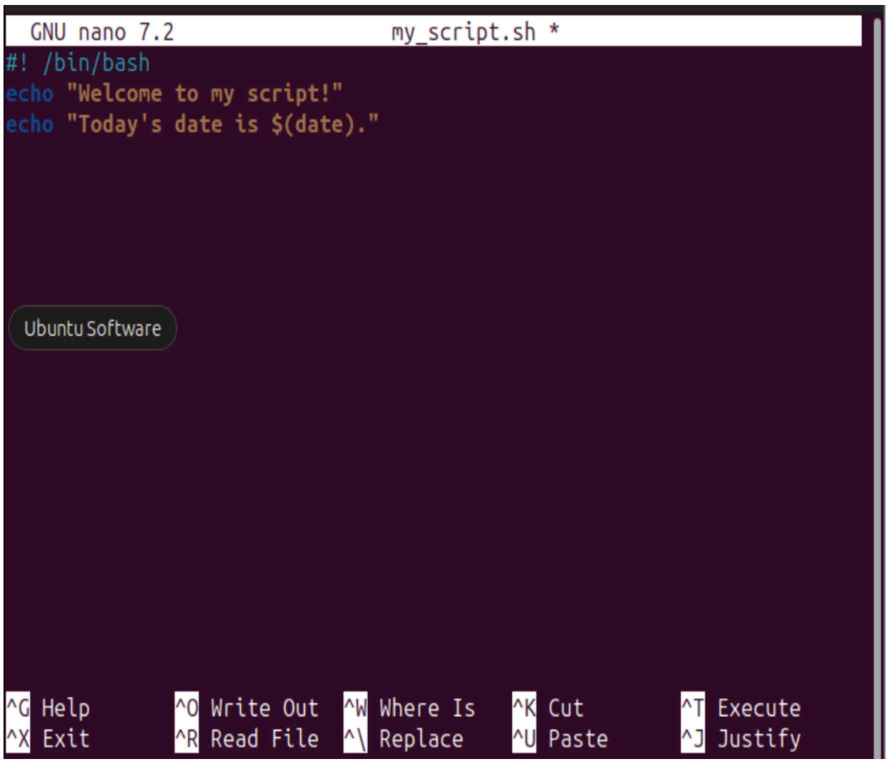
echo "Today's date is $(date)."

Save and exit the file.



```
ubuntu@ubuntu:~/my_directory$ nano my_script.sh
```

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

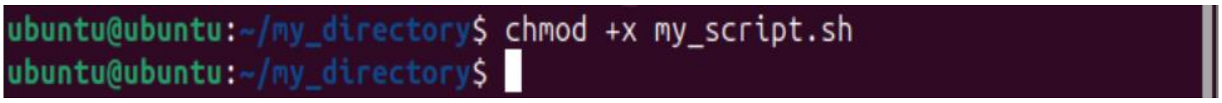


```
  GNU nano 7.2                    my_script.sh *
#! /bin/bash
echo "Welcome to my script!"
echo "Today's date is $(date)."

^G Help       ^O Write Out  ^W Where Is   ^K Cut        ^T Execute
^X Exit       ^R Read File  ^\ Replace    ^U Paste      ^J Justify
```

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.**

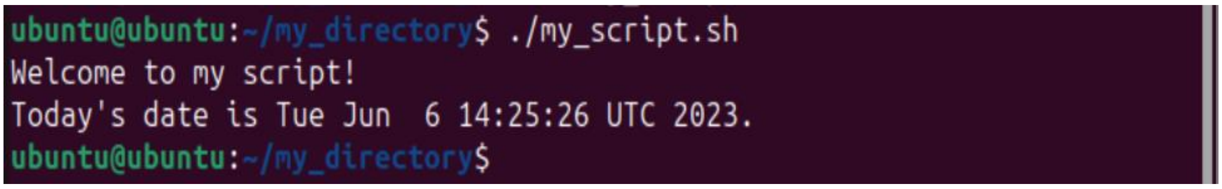
- Make "my_script.sh" executable



```
ubuntu@ubuntu:~/my_directory$ chmod +x my_script.sh
ubuntu@ubuntu:~/my_directory$ 
```

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.

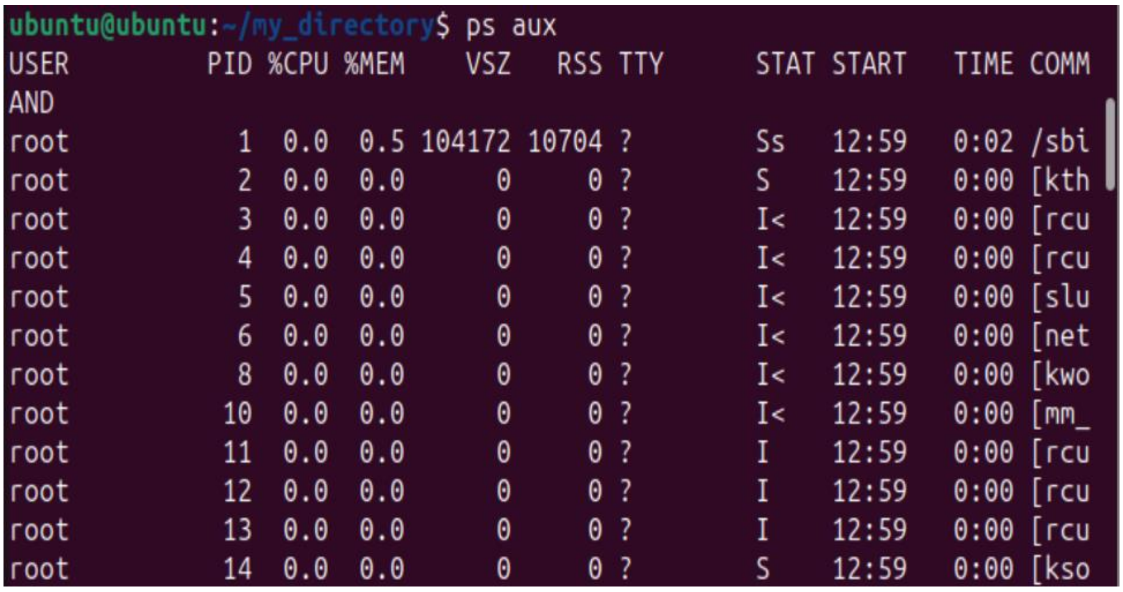


```
ubuntu@ubuntu:~/my_directory$ ./my_script.sh
Welcome to my script!
Today's date is Tue Jun  6 14:25:26 UTC 2023.
ubuntu@ubuntu:~/my_directory$
```

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

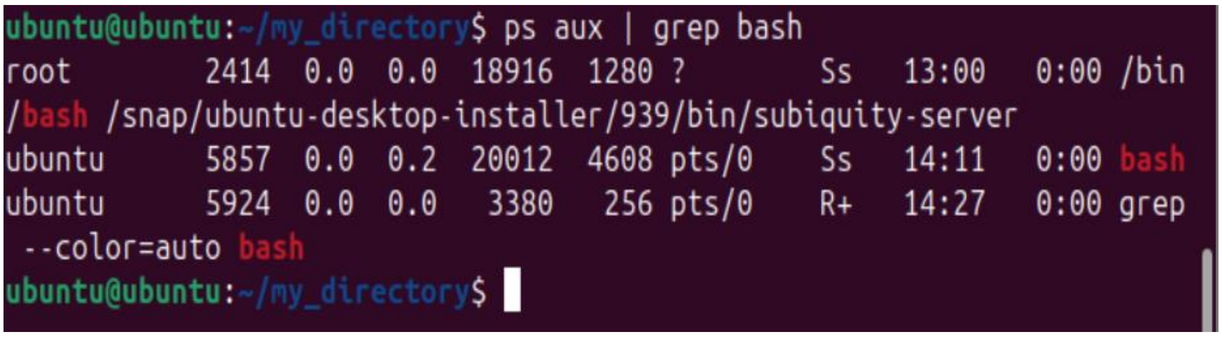
- List all the processes running on your system using the "ps" command.



```
ubuntu@ubuntu:~/my_directory$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMM
AND
root           1  0.0  0.5 104172 10704 ?        Ss   12:59   0:02 /sbi
root           2  0.0  0.0      0     0 ?        S    12:59   0:00 [kth
root           3  0.0  0.0      0     0 ?        I<   12:59   0:00 [rcu
root           4  0.0  0.0      0     0 ?        I<   12:59   0:00 [rcu
root           5  0.0  0.0      0     0 ?        I<   12:59   0:00 [slu
root           6  0.0  0.0      0     0 ?        I<   12:59   0:00 [net
root           8  0.0  0.0      0     0 ?        I<   12:59   0:00 [kwo
root          10  0.0  0.0      0     0 ?        I<   12:59   0:00 [mm_
root          11  0.0  0.0      0     0 ?        I    12:59   0:00 [rcu
root          12  0.0  0.0      0     0 ?        I    12:59   0:00 [rcu
root          13  0.0  0.0      0     0 ?        I    12:59   0:00 [rcu
root          14  0.0  0.0      0     0 ?        S    12:59   0:00 [kso
```

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.**

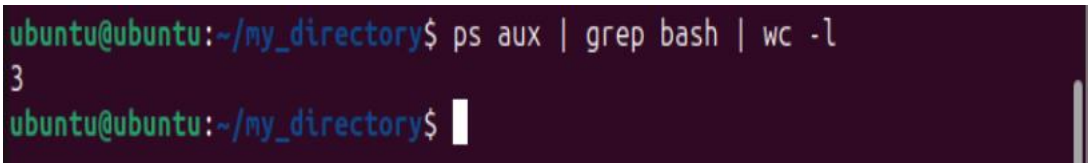
- Use the "grep" command to filter the processes list and display only the processes with "bash" in their name.



```
ubuntu@ubuntu:~/my_directory$ ps aux | grep bash
root         2414  0.0  0.0  18916  1280 ?        Ss   13:00   0:00 /bin
/bash /snap/ubuntu-desktop-installer/939/bin/subiquity-server
ubuntu       5857  0.0  0.2  20012  4608 pts/0    Ss   14:11   0:00 bash
ubuntu       5924  0.0  0.0   3380   256 pts/0    R+   14:27   0:00 grep
 --color=auto bash
ubuntu@ubuntu:~/my_directory$
```

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.



```
ubuntu@ubuntu:~/my_directory$ ps aux | grep bash | wc -l
3
ubuntu@ubuntu:~/my_directory$
```

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.**