

Assignment 2: Bash Shell Basics

Name: Gitesh Bhavsar

Date: 28-05-2023

Regs_No: 20BCR7083

Campus: VIT-AP

Task 1: File and Directory Manipulation

1. Create a directory called "my_directory".

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

2. Navigate into the "my_directory".

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

3. Create an empty file called "my_file.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.

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

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

```
File Actions Edit View Help
(kali㉿kali)-[~]
$ cd Desktop

(kali㉿kali)-[~/Desktop]
$ mkdir my_directory

(kali㉿kali)-[~/Desktop]
$ cd my_directory

(kali㉿kali)-[~/Desktop/my_directory]
$ touch my_file.txt

(kali㉿kali)-[~/Desktop/my_directory]
$ ls
my_file.txt

(kali㉿kali)-[~/Desktop/my_directory]
$ mv my_file.txt new_file.txt

(kali㉿kali)-[~/Desktop/my_directory]
$ ls
new_file.txt

(kali㉿kali)-[~/Desktop/my_directory]
$ less new_file.txt

(kali㉿kali)-[~/Desktop/my_directory]
$
```

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

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".

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".

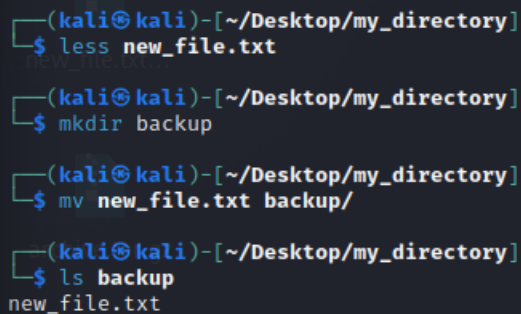
This command creates a new directory named "backup" within the "my_directory" directory.

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

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



```
(kali㉿kali)-[~/Desktop/my_directory]
$ less new_file.txt

(kali㉿kali)-[~/Desktop/my_directory]
$ mkdir backup

(kali㉿kali)-[~/Desktop/my_directory]
$ mv new_file.txt backup/

(kali㉿kali)-[~/Desktop/my_directory]
$ ls backup
new_file.txt
```

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

```
(kali㉿kali)-[~/Desktop/my_directory]
$ mkdir backup

(kali㉿kali)-[~/Desktop/my_directory]
$ mv new_file.txt backup/

(kali㉿kali)-[~/Desktop/my_directory]
$ ls backup
new_file.txt

(kali㉿kali)-[~/Desktop/my_directory]
$ rm -r backup/

(kali㉿kali)-[~/Desktop/my_directory]
$ ls

(kali㉿kali)-[~/Desktop/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

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

```
(kali㉿kali)-[~/Desktop/my_directory]
$ touch my_script.sh

(kali㉿kali)-[~/Desktop/my_directory]
$ nano my_script.sh
```

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

2. 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.

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

```
kali@kali: ~/Desktop/my_directory
File Actions Edit View Help
GNU nano 7.2 my_script.sh
#!/bin/bash
echo "Welcome to my script!"
echo "Today's date is $(date)."
```

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.

3. Make "my_script.sh" executable

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.

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

```
(kali㉿kali)-[~/Desktop/my_directory]
$ chmod +x my_script.sh

(kali㉿kali)-[~/Desktop/my_directory]
$ ./my_script.sh
Welcome to my script!
Today's date is Sun May 28 12:50:08 PM EDT 2023.

(kali㉿kali)-[~/Desktop/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

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

```
(kali㉿kali)-[~/Desktop/my_directory]
$ ps aux
USER          PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root            1  0.0  0.5 167764 12116 ?        Ss   12:20   0:00 /sbin/init splash
root            2  0.0  0.0      0     0 ?        S    12:20   0:00 [kthreadd]
root            3  0.0  0.0      0     0 ?        I<   12:20   0:00 [rcu_gp]
root            4  0.0  0.0      0     0 ?        I<   12:20   0:00 [rcu_par_gp]
root            5  0.0  0.0      0     0 ?        I<   12:20   0:00 [slub_flushwq]
root            6  0.0  0.0      0     0 ?        I<   12:20   0:00 [netns]
root            8  0.0  0.0      0     0 ?        I<   12:20   0:00 [kworker/0:0H-events_highpri]
root            9  0.0  0.0      0     0 ?        I    12:20   0:00 [kworker/u4:0-events_unbound]
root           10  0.0  0.0      0     0 ?        I<   12:20   0:00 [mm_percpu_wq]
root           11  0.0  0.0      0     0 ?        I    12:20   0:00 [rcu_tasks_kthread]
root           12  0.0  0.0      0     0 ?        I    12:20   0:00 [rcu_tasks_rude_kthread]
root           13  0.0  0.0      0     0 ?        I    12:20   0:00 [rcu_tasks_trace_kthread]
root           14  0.0  0.0      0     0 ?        S    12:20   0:00 [ksoftirqd/0]
root           15  0.0  0.0      0     0 ?        I    12:20   0:01 [rcu_preempt]
root           16  0.0  0.0      0     0 ?        S    12:20   0:00 [migration/0]
root           18  0.0  0.0      0     0 ?        S    12:20   0:00 [cpuhp/0]
root           19  0.0  0.0      0     0 ?        S    12:20   0:00 [cpuhp/1]
root           20  0.0  0.0      0     0 ?        S    12:20   0:00 [migration/1]
root           21  0.0  0.0      0     0 ?        S    12:20   0:00 [ksoftirqd/1]
root           26  0.0  0.0      0     0 ?        S    12:20   0:00 [kdevtmpfs]
root           27  0.0  0.0      0     0 ?        I<   12:20   0:00 [inet_frag_wq]
root           28  0.0  0.0      0     0 ?        S    12:20   0:00 [kauditd]
root           29  0.0  0.0      0     0 ?        S    12:20   0:00 [khungtaskd]
root           30  0.0  0.0      0     0 ?        S    12:20   0:00 [oom_reaper]
root           32  0.0  0.0      0     0 ?        I<   12:20   0:00 [writeback]
root           33  0.0  0.0      0     0 ?        S    12:20   0:00 [kcompactd0]
root           34  0.0  0.0      0     0 ?        SN   12:20   0:00 [ksmd]
root           35  0.0  0.0      0     0 ?        SN   12:20   0:00 [khugepaged]
root           36  0.0  0.0      0     0 ?        I<   12:20   0:00 [kintegrityd]
root           37  0.0  0.0      0     0 ?        I<   12:20   0:00 [kblockd]
root           38  0.0  0.0      0     0 ?        I<   12:20   0:00 [blkcg_punt_bio]
root           39  0.0  0.0      0     0 ?        I<   12:20   0:00 [tpm_dev_wq]
root           40  0.0  0.0      0     0 ?        I<   12:20   0:00 [edac-poller]
root           41  0.0  0.0      0     0 ?        I<   12:20   0:00 [devfreq_wq]
root           42  0.1  0.0      0     0 ?        I    12:20   0:03 [kworker/1:1-events]
root           43  0.0  0.0      0     0 ?        I<   12:20   0:00 [kworker/0:1H-kblockd]
root           44  0.0  0.0      0     0 ?        S    12:20   0:00 [kswapd0]
root           50  0.0  0.0      0     0 ?        I<   12:20   0:00 [kthrotld]
root           52  0.0  0.0      0     0 ?        I<   12:20   0:00 [acpi_thermal_pm]
root           53  0.0  0.0      0     0 ?        S    12:20   0:00 [xenbus_probe]
root           54  0.0  0.0      0     0 ?        I<   12:20   0:00 [mld]
root           55  0.0  0.0      0     0 ?        I<   12:20   0:00 [ipv6_addrconf]
root           60  0.0  0.0      0     0 ?        I<   12:20   0:00 [kstrp]
root           65  0.0  0.0      0     0 ?        I<   12:20   0:00 [zswap-shrink]
root           66  0.0  0.0      0     0 ?        I<   12:20   0:00 [kworker/u5:0]
root          130  0.0  0.0      0     0 ?        I<   12:20   0:00 [kworker/1:1H-kblockd]
root          134  0.0  0.0      0     0 ?        I<   12:20   0:00 [cryptd]
```

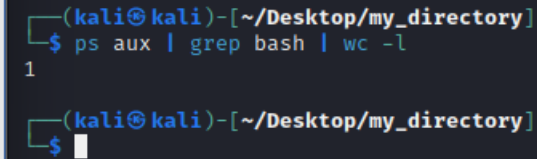
The ps command is used to display information about active processes. The aux options provide a detailed list of all processes running on the system.

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

```
(kali㉿kali)-[~/Desktop/my_directory]
$ ps aux | grep bash
kali          16545  0.0  0.1  6332  2052 pts/0    S+   12:50   0:00 grep --color=auto bash
(kali㉿kali)-[~/Desktop/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"

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



```
(kali㉿kali)-[~/Desktop/my_directory]
$ ps aux | grep bash | wc -l
1

(kali㉿kali)-[~/Desktop/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.