EXPERIMENT 7 - Shell Programming

Aim

To understand and practically apply process management commands, viewing and monitoring processes, terminating, process priortization and scheduling

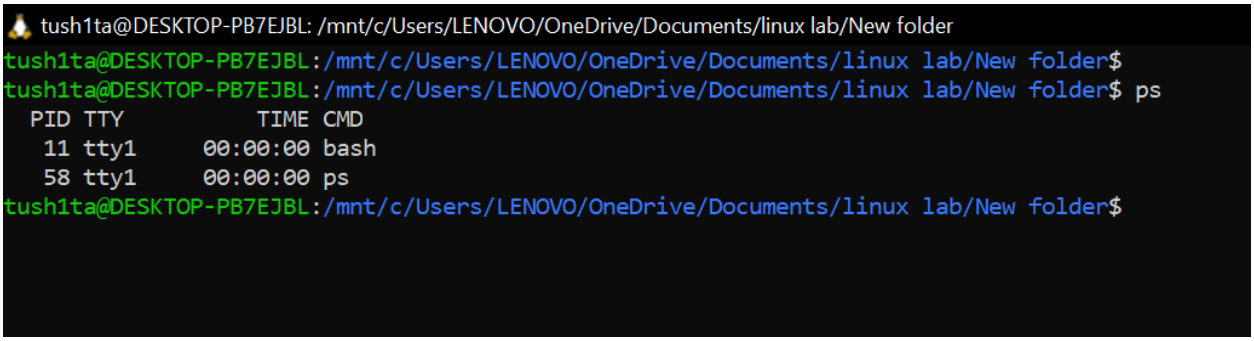
---

Tools & Software Used

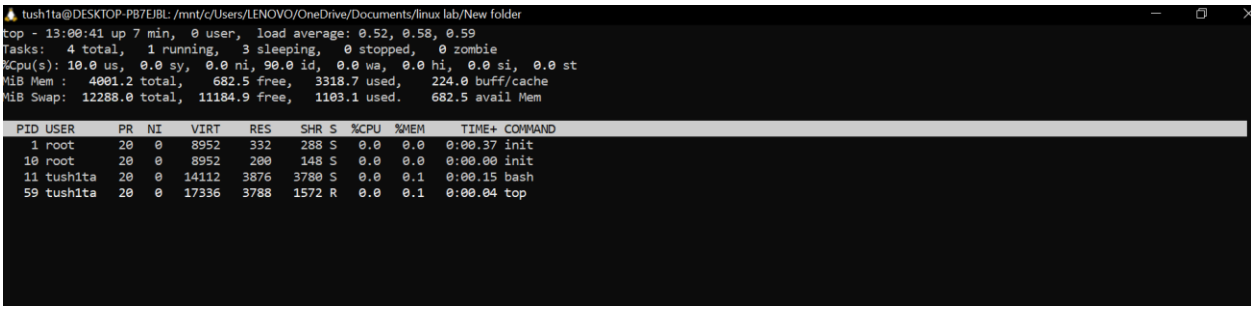
- **Terminal Emulator:** GNOME Terminal
- **Shell:** Bash (*Bourne-Again Shell*)

Basic Process Commands

1. `ps` - shows currently running processes



2. `top` - shows running processes dynamically with their CPU and memory usage



3. `pstree` - shows process hierarchy

```
tush1ta@DESKTOP-PB7EJBL: /mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder

tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder$ pstree
init─┬─init───bash───pstree
     └─{init}
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder$
```

4. `kill<PID>` - stops the process

```
tush1ta@DESKTOP-PB7EJBL: /mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder$ sleep 30 &
[1] 63
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder$ ps
  PID TTY          TIME CMD
   11 tty1     00:00:00 bash
   63 tty1     00:00:00 sleep
   64 tty1     00:00:00 ps
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder$ kill 63
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder$ ps
  PID TTY          TIME CMD
   11 tty1     00:00:00 bash
   65 tty1     00:00:00 ps
[1]+  Terminated              sleep 30
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder$
```

Process Priotization

1. `nice -n <value> command` - starts a process with a specific priority

```
tush1ta@DESKTOP-PB7EJBL: /mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder$ nice -n 15 sleep 600 &
[1] 67
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder$ nice -n 13 sleep 601 &
[2] 68
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder$ nice -n 17 sleep 602 &
[3] 69
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder$ ps -o pid,comm,ni,pri -p $(pgrep -d, -f "sleep")
  PID COMMAND         NI PRI
   67 sleep          -15  34
   68 sleep          -13  32
   69 sleep          -17  36
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder$
```

2. `renice <value> - p <PID>` - change priority of a runnung process

```
tush1ta@DESKTOP-PB7EJBL: /mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder
   69 sleep          -17  36
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder$ nice -n 14 sleep 180 &
[4] 72
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder$ renice 15 -p 72
72 (process ID) old priority 14, new priority 15
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder$
```

Process Scheduling

1. *at <TIME> - schedules one time tasks (works only with mins, hours & days)*

2. *cron - runs specific tasks at given time / dates*

-
  -
    -
      - command │ │ │ │ │ │ │ │ └── Day of week (0–6, Sunday=0) │ │ │ └──── Month (1–12) │ │ └────── Day of month (1–31) │ └──────── Hour (0–23) └────────── Minute (0–59)

LAB Exericeses

TASK 1: File existence Check

Script:

```
tush1ta@DESKTOP-PB7EJBL: /mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp7
#!/bin/bash
echo "Enter filename "
read file

if [ -e "$file" ]
then
  echo "File exists. Contents are:"
  cat "$file"
else
echo "File does not exist"
echo "Do you want to create it? (y/n)"
read choice
if [ "$choice" = "y" ]; then
touch "$file"
echo "File $file created."
fi
```

Output:

```
tush1ta@DESKTOP-PB7EJBL: /mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp7
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp7$ ./Task7.1
Enter filename
file1
File does not exist
Do you want to create it? (y/n)
y
File file1 created.
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp7$
```

TASK 2: Print no.'s from 1 to 10

Output

```
tush1ta@DESKTOP-PB7EJBL: /mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp7
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp7$ vim Task7.2
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp7$ ./Task7.2
1
2
3
4
5
6
7
8
9
10
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp7$
```

TASK 3: Count lines, words and characters in a file

Script:

```
tush1ta@DESKTOP-PB7EJBL: /mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp7
#!/bin/bash
if [ $# -eq 0 ]
then
echo "Usage: $0 filename"
exit 1
fi
file=$1
if [ -e "$file" ]
then
echo "Lines: $(wc -l < $file)"
echo "Words; $(wc -w < $file)"
echo "Characters:  $(wc -m <$file)"
else
echo "file not found"
fi
~
~
~
```

Output:

```
tush1ta@DESKTOP-PB7EJBL: /mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp7
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp7$ ./Task7.3 file1
Lines: 0
Words; 0
Characters:  0
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp7$
```

TASK 4: Factorial of a number using funtion

Script:

```
tush1ta@DESKTOP-PB7EJBL: /mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp7
#!/bin/bash
factorial() {
    num=$1
        fact=1
            while [ "$num" -gt 1 ]
                do
                        fact=$((fact * num))
                                num=$((num - 1))
                                    done
                                                                                echo $fact
                                        }

                                        echo "Factorial of 5 is $(factorial 5)"
                                        echo "Factorial of 7 is $(factorial 7)"
                                        echo "Factorial of 10 is $(factorial 10)"
~
~
~
```

Output:

```
tush1ta@DESKTOP-PB7EJBL: /mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp7
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp7$ ./Task7.4
Factorial of 5 is 120
Factorial of 7 is 5040
Factorial of 10 is 3628800
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp7$
```

OBSERVATIONS

- Successfully viewed running processes using ps, top, and pstree.
- Able to terminate processes using kill and control their priority with nice and renice.
- Scheduled tasks using at for one-time execution and cron for recurring jobs.

CONCLUSION

The experiment enhanced understanding of Linux process management and scheduling.

EXPERIMENT 8 - Shell Programming

Name: Tushita Sharma SAP id:590029121 Date:29/09/2025

Aim

To understand and demonstrate the concepts of process control and signals, process monitoring and resource usage, process communication and synchronization, background processes and job control, and system monitoring and logging in Linux

---

Tools & Software Used

- **Operating System:** Ubuntu running on Oracle VirtualBox
- **Terminal Emulator:** GNOME Terminal
- **Shell:** Bash (*Bourne-Again Shell*)

Process Control & signals

- `kill` - used to terminate or send signals to processes

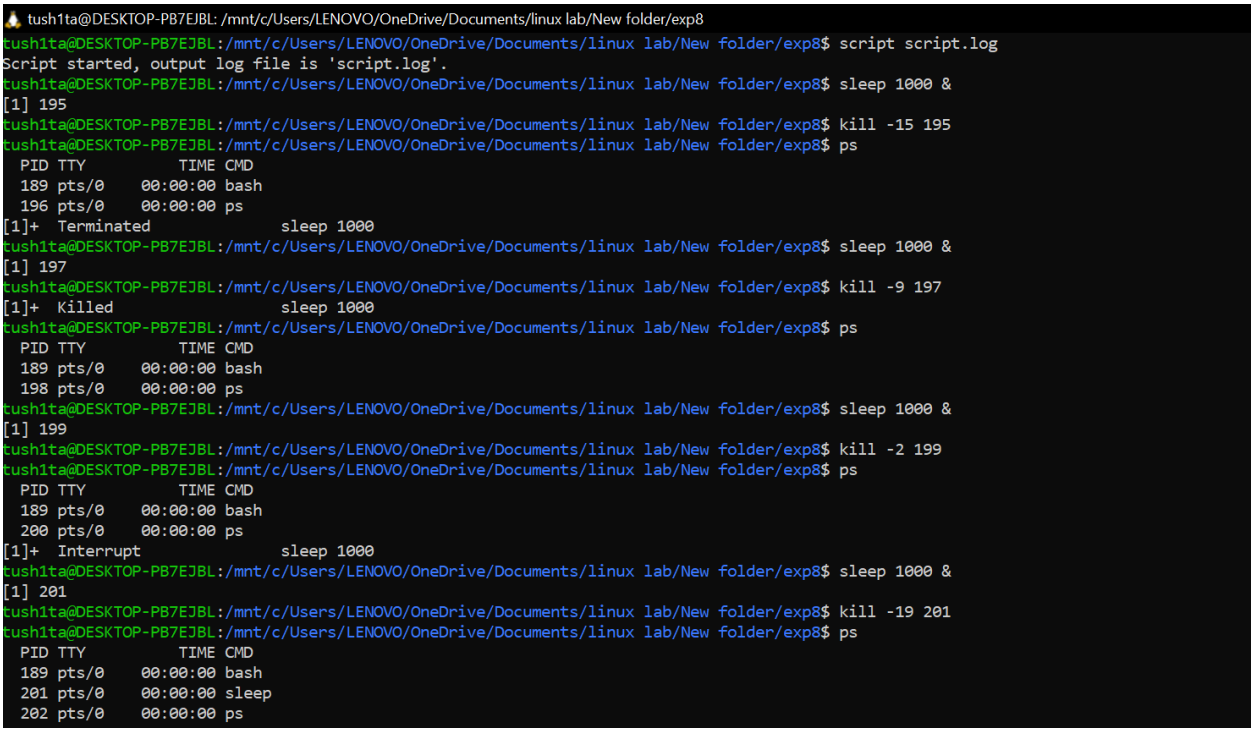
Syntax:

```
kill [options] <PID>
```

Signals (used as options):

1. `2` - **SIGINT** (*interrupt*)
2. `15` - **SIGTERM** (*termiante gracefully*)
3. `9` - **SIGKILL** (*force kill*)
4. `19` - **SIGSTOP** (*Stops a process*)
5. `18` - **SIGCONT** (*resumes the stopped process*)

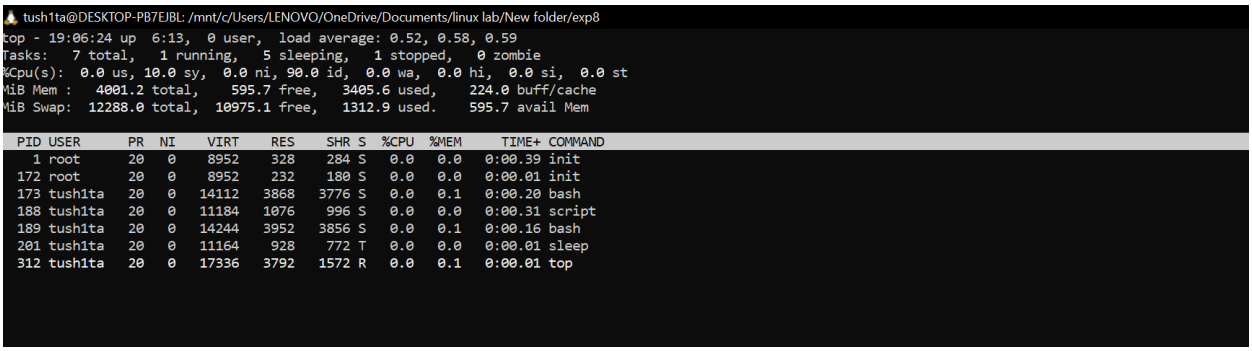
OUTPUT:



Process Monitoring and Resource Usage

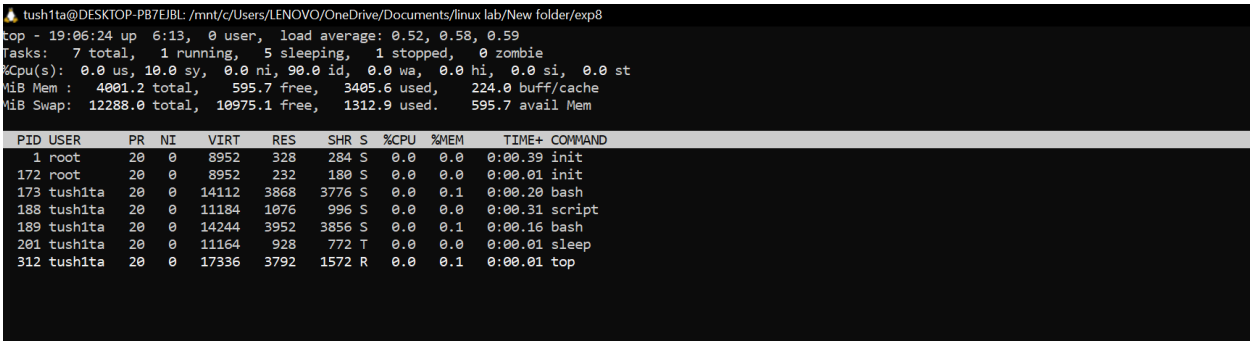
1. `top` - **live view of processes, CPU, memory**

OUTPUT:



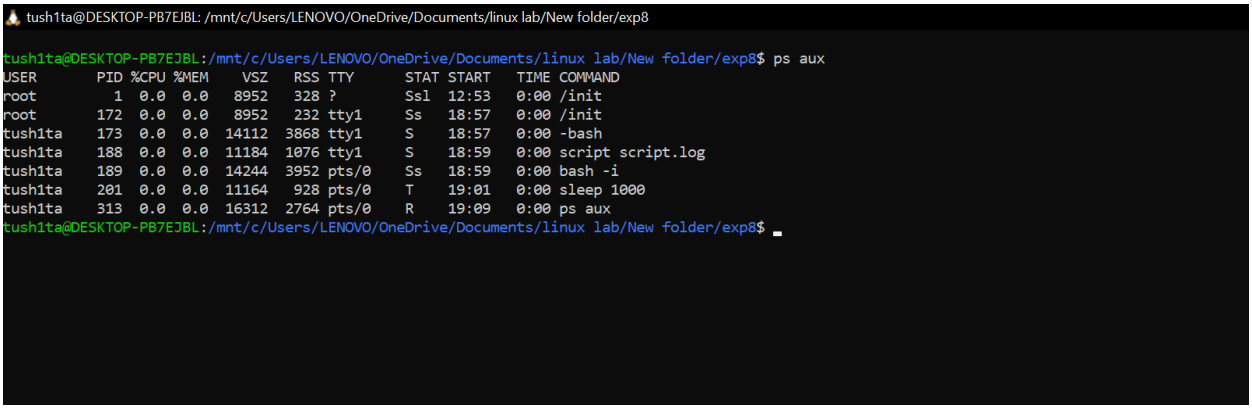
2. `htop` - **user friendly version of *top***

OUTPUT:

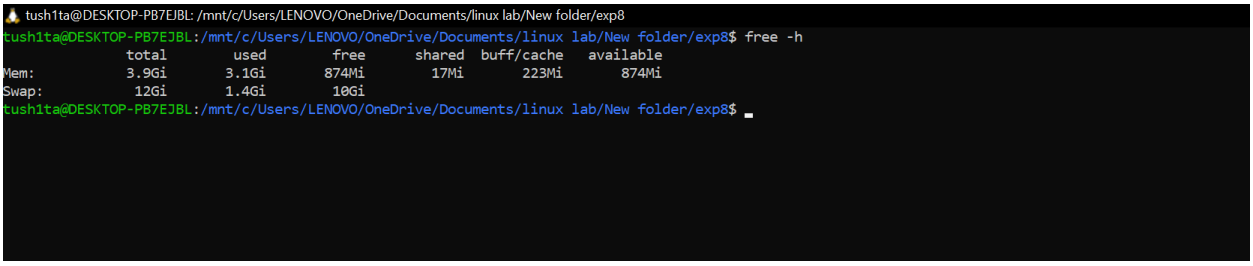


3. `ps aux` - **snapshot of all processes**

OUTPUT:



4. `free -h` - **show memory usage**



5. `uptime` - **system load averages**

OUTPUT:

Process Communication

- **Pipes | - to pass output of one command to another**

OUTPUT:

```
tush1ta@DESKTOP-PB7EJBL: /mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp8
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp8$ ps aux | grep bash
tush1ta     173  0.0  0.0  14112  3868 tty1     S    18:57   0:00 -bash
tush1ta     189  0.0  0.1  14244  4160 pts/0    Ss   18:59   0:00 bash -i
tush1ta     318  0.0  0.0  12128  1304 pts/0    S    19:12   0:00 grep --color=auto bash
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp8$
```

Process Synchronization

To prevent conflicts, processes can be synchronized

- `wait` - **used to pause the execution of a script until all the background processes complete.**

Script:

```
tush1ta@DESKTOP-PB7EJBL: /mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp8
#!/bin/bash
 echo " Starting Processes"
 nice -n 4 sleep 5 &
 PID1=$!
 wait
 echo " sleep 5 is completed with PID = $PID1 "

 nice -n 6 sleep 3 &
 PID2=$!
 wait
echo " sleep 3 is completed with PID = $PID3 "
echo "ALL processes are completed"

#end
~
~
~
```

Output:

```
tush1ta@DESKTOP-PB7EJBL: /mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp8
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp8$ vim task8.sh
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp8$ ./task8.sh
 Starting Processes
 sleep 5 is completed with PID = 329
 sleep 3 is completed with PID =
ALL processes are completed
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp8$
```

- `wait <PID>` - **waits for a particular job to finish**

SCRIPT:

```
tush1ta@DESKTOP-PB7EJBL: /mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp8
#!/bin/bash
 echo " Starting Processes"
 nice -n 4 sleep 7 &
 PID1=$!
 echo "Started sleep 5 with PID = $PID1 "

 nice -n 6 sleep 3 &
 PID2=$!
 echo "Started sleep 3 with PID = $PID2"


 nice -n 7 sleep 10 &
PID3=$!
echo " Started  sleep 7 with PID = $PID3"

wait $PID2


echo "Processes with PID $PID2 completed"
echo "Processes with PIDs $PID1 and $PID3 are still running in the background"
#end
~
~
~
```

OUTPUT:

```
tush1ta@DESKTOP-PB7EJBL: /mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp8
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp8$ ./task8.sh
 Starting Processes
Started sleep 5 with PID = 346
Started sleep 3 with PID = 347
 Started  sleep 7 with PID = 348
Processes with PID 347 completed
Processes with PIDs 346 and 348 are still running in the background
tush1ta@DESKTOP-PB7EJBL:/mnt/c/Users/LENOVO/OneDrive/Documents/linux lab/New folder/exp8$
```

Background Process and Job control

1. `&` - **used to run a process in background**
2. `jobs` - **shows background jobs**
3. `fg %1` - **brings job 1 to foreground**
4. `bg %1` - **resume job 1 in background**

OUTPUT:

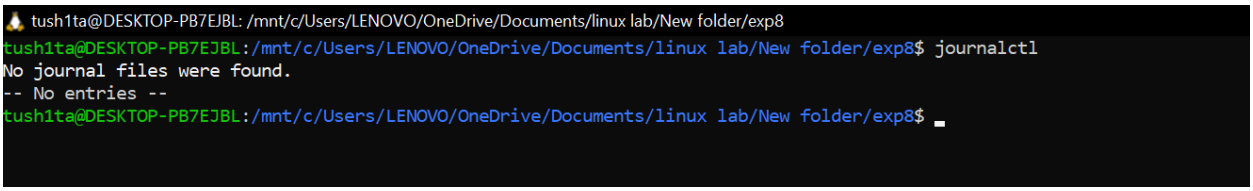
System Monitoring and logging

1. `dmesg | less` - **kernel/ system messages**

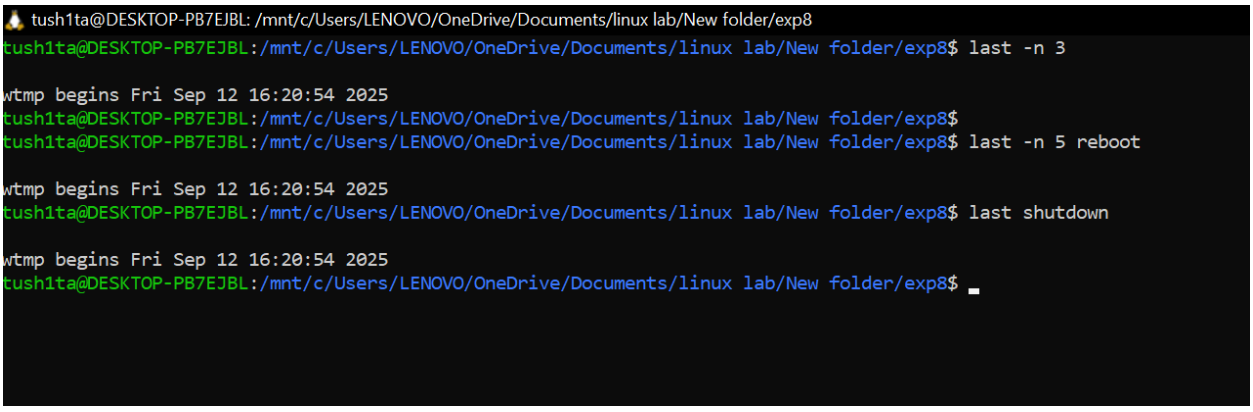
OUTPUT:

2. `journalctl` - **systemlogs

   OUTPUT:

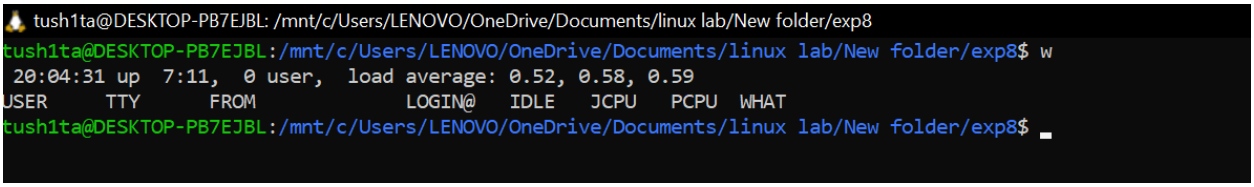


3. `Last – Logged-in users`



4. `who` or `w` - **user currently logged-in**

OUTPUT:



LAB Exericeses

TASK 1: Check File Permissions

Write a script that checks the file permissions of a given file and displays whether it is readable, writable, or executable by the current user.

TASK 2: String Operations

Create a script that prompts the user to enter a string and then performs operations like string length, string concatenation, and string comparison.

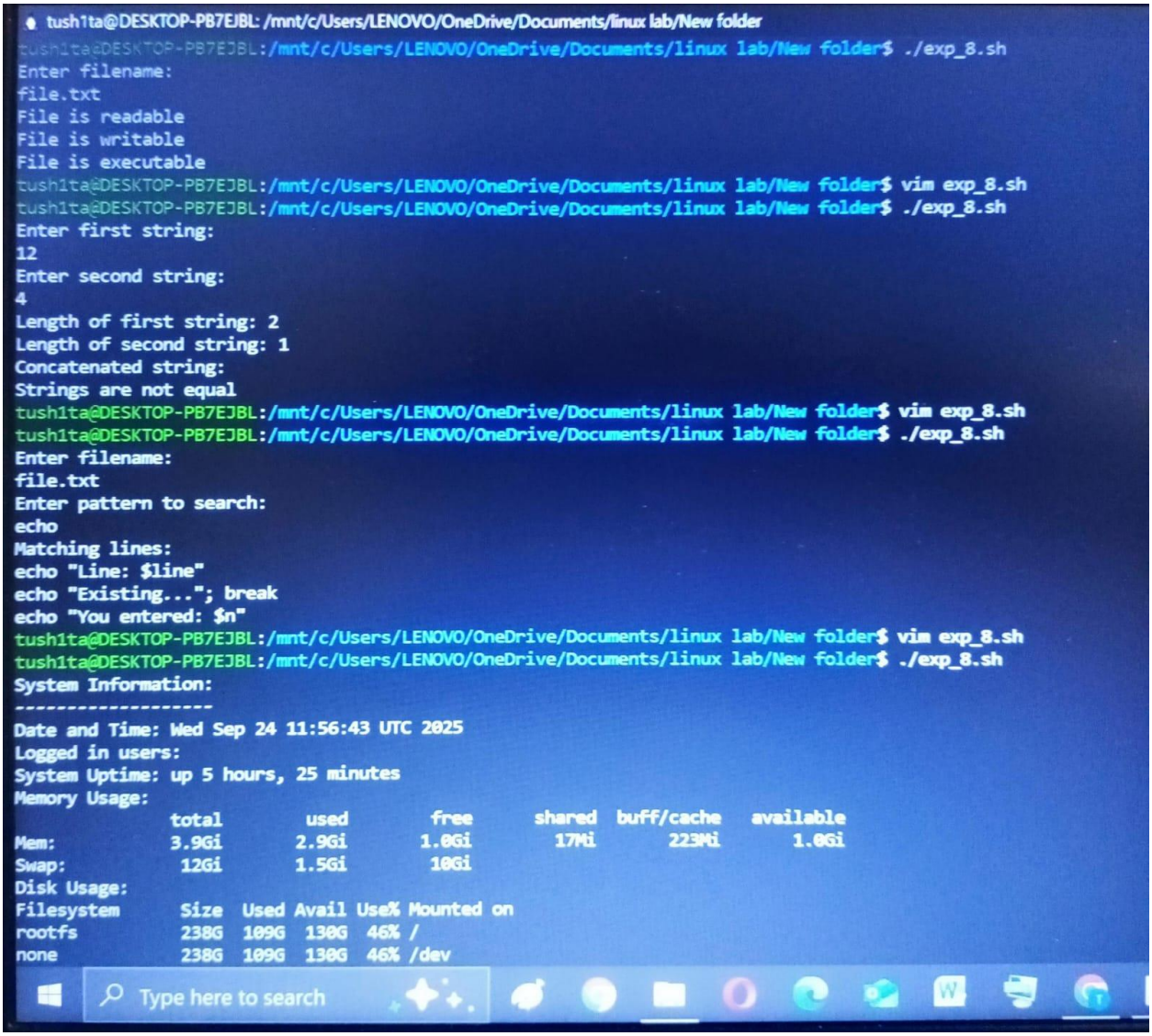
TASK 3: Search for a Pattern in a file

Write a script that searches for a specific pattern in a given file and displays the matching lines.

TASK 4: Display System Information

Create a script that displays various system information like the current date and time, logged-in users, system uptime, etc.

Output:



OBSERVATIONS

- `kill`, `wait` and job control commands (`&`, `jobs`, `fg`, `bg`) worked as expected.
- `top`, `htop`, `ps aux`, and `free -h` provided real-time process and resource information.
- Pipes (`|`) enabled inter-process communication.
- System monitoring commands (`dmesg`, `journalctl`, `last`, `who`) displayed logs and user activity correctly.
- Lab exercises executed successfully with expected outputs.

CONCLUSION

- The experiment demonstrated process control, monitoring, communication, and synchronization in Linux.
- Background job management and system monitoring help efficiently manage processes.
- Shell scripting with process commands enables effective automation and resource tracking.