590029302_Exp[7]_ScriptLog

Experiment 7: Shell Programming, Process and Scheduling

Name: Tanmay Amit Verma Roll No.: 590029302 Date: 2025-09-23

Aim:

- To write shell scripts that demonstrate process management.
- To understand how to schedule processes using `cron` and `at`.
- To monitor running processes and practice job control commands.

Requirements

- A Linux machine with bash shell.
- Access to process management commands (`ps`, `top`, `kill`, `jobs`, `fg`, `bg`).
- Access to scheduling utilities (`cron`, `at`).

Theory

Every program running in Linux is a process identified by a unique process ID (PID). Shell programming allows automation of tasks including spawning and controlling processes. Process management commands like `ps`, `top`, `kill`, `jobs`, `bg`, and `fg` let users monitor and control execution. Scheduling utilities such as `cron` (repeated tasks) and `at` (one-time tasks) allow tasks to run automatically at defined times. Combining scripting with scheduling is a core system administration skill.

Procedure & Observations

Task 1

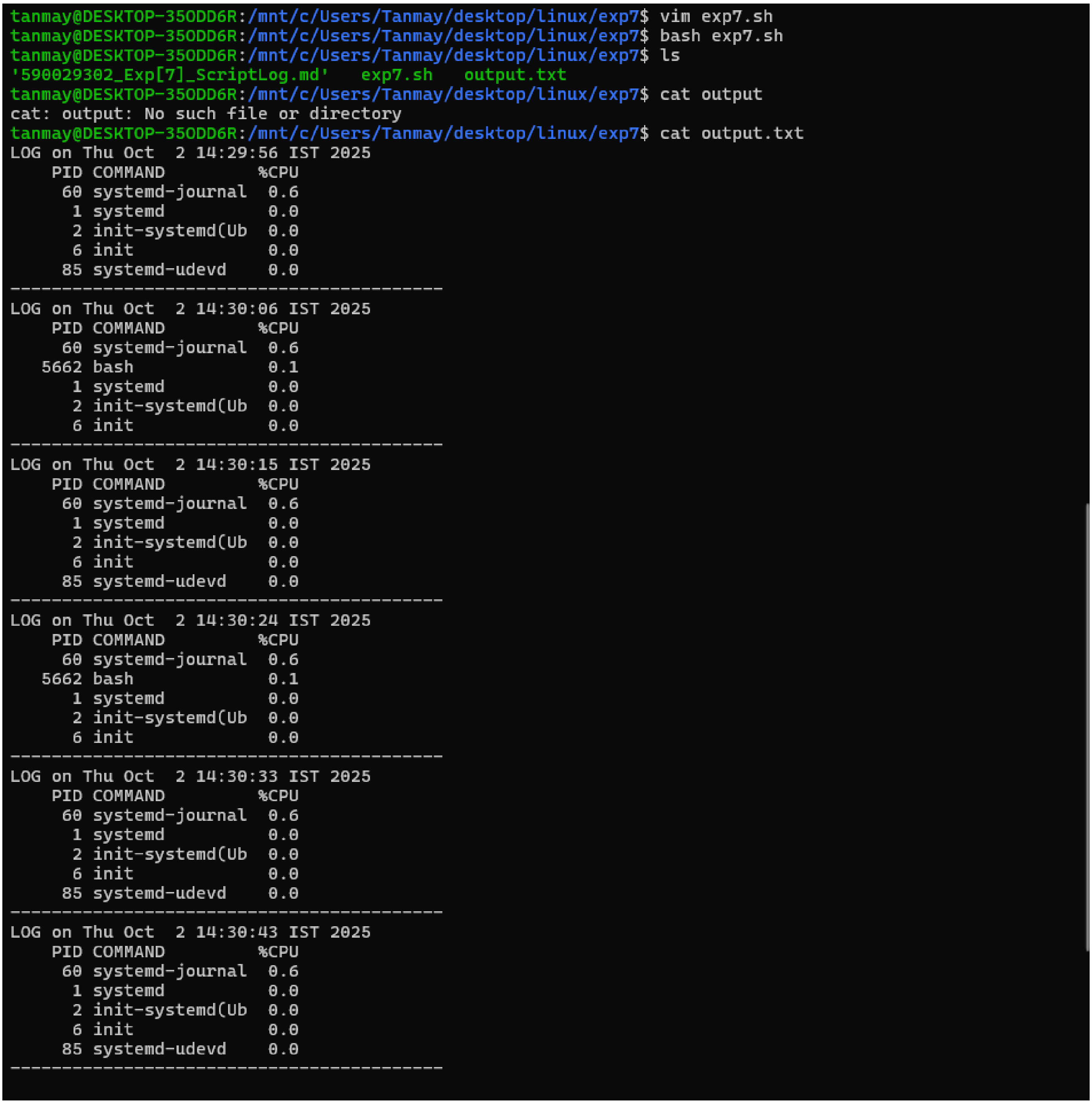
Task Statement:

Write a script that monitors the top 5 processes consuming the most CPU and logs them into a file every 10 seconds.

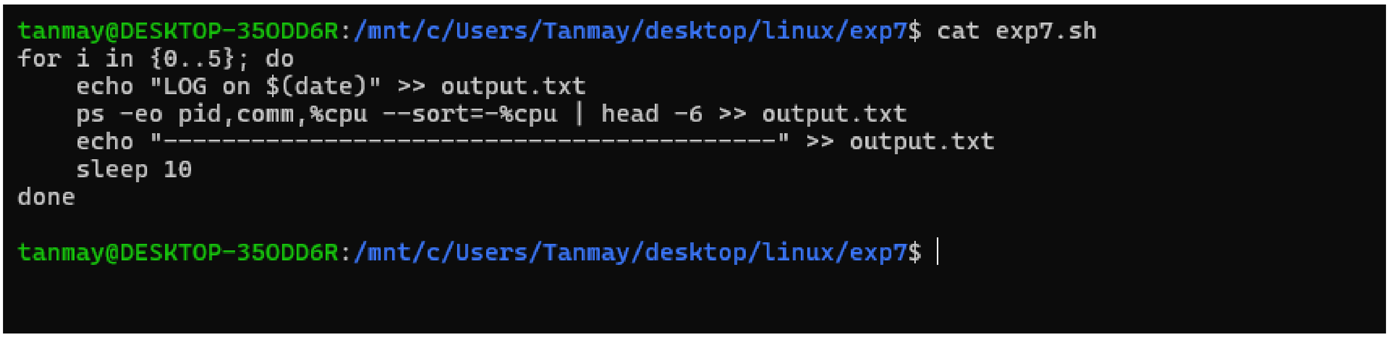
Command(s):

```
for i in {0..5}; do
    echo "LOG on $(date)" >> output.txt
    ps -eo pid,comm,%cpu --sort=-%cpu | head -6 >> output.txt
    echo "-------------------------------------------" >> output.txt
    sleep 10
done
```

Output:



```
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ vim exp7.sh
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ bash exp7.sh
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ ls
'590029302_Exp[7]_ScriptLog.md'   exp7.sh   output.txt
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ cat output
cat: output: No such file or directory
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ cat output.txt
LOG on Thu Oct  2 14:29:56 IST 2025
    PID COMMAND         %CPU
     60 systemd-journal  0.6
      1 systemd          0.0
      2 init-systemd(Ub  0.0
      6 init             0.0
     85 systemd-udevd    0.0
------------------------------------------
LOG on Thu Oct  2 14:30:06 IST 2025
    PID COMMAND         %CPU
     60 systemd-journal  0.6
   5662 bash             0.1
      1 systemd          0.0
      2 init-systemd(Ub  0.0
      6 init             0.0
------------------------------------------
LOG on Thu Oct  2 14:30:15 IST 2025
    PID COMMAND         %CPU
     60 systemd-journal  0.6
      1 systemd          0.0
      2 init-systemd(Ub  0.0
      6 init             0.0
     85 systemd-udevd    0.0
------------------------------------------
LOG on Thu Oct  2 14:30:24 IST 2025
    PID COMMAND         %CPU
     60 systemd-journal  0.6
   5662 bash             0.1
      1 systemd          0.0
      2 init-systemd(Ub  0.0
      6 init             0.0
------------------------------------------
LOG on Thu Oct  2 14:30:33 IST 2025
    PID COMMAND         %CPU
     60 systemd-journal  0.6
      1 systemd          0.0
      2 init-systemd(Ub  0.0
      6 init             0.0
     85 systemd-udevd    0.0
------------------------------------------
LOG on Thu Oct  2 14:30:43 IST 2025
    PID COMMAND         %CPU
     60 systemd-journal  0.6
      1 systemd          0.0
      2 init-systemd(Ub  0.0
      6 init             0.0
     85 systemd-udevd    0.0
------------------------------------------
```



```
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ cat exp7.sh
for i in {0..5}; do
    echo "LOG on $(date)" >> output.txt
    ps -eo pid,comm,%cpu --sort=-%cpu | head -6 >> output.txt
    echo "------------------------------------------" >> output.txt
    sleep 10
done

tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ 
```

Task 2

Task Statement:

Write a script that accepts a PID from the user and displays its details (state, parent process, memory usage).

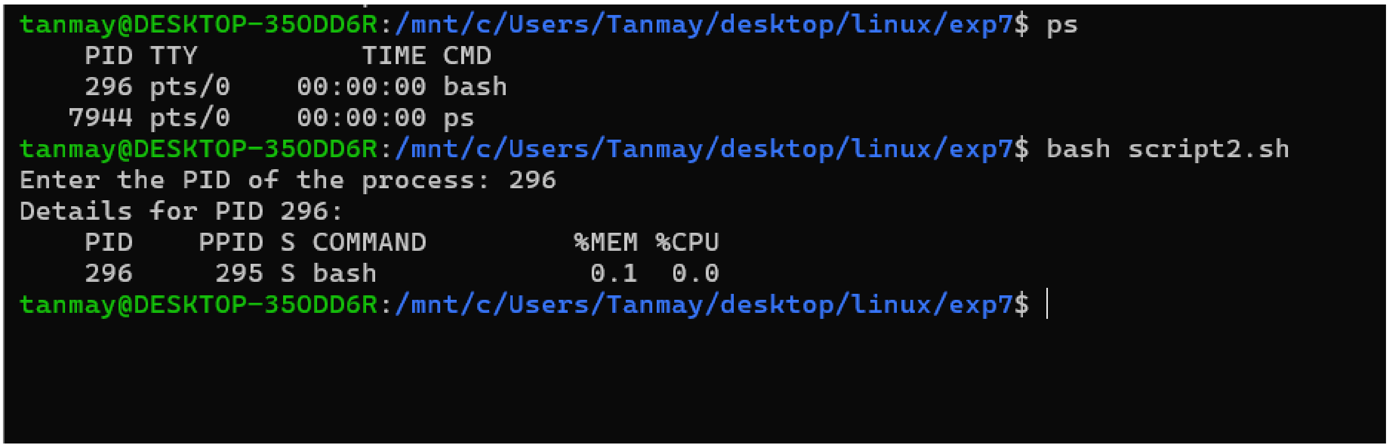
Command(s):

```
#!/bin/bash

read -p "Enter the PID of the process: " pid

echo "Details for PID $pid:"
ps -p "$pid" -o pid,ppid,state,comm,%mem,%cpu
```

Output:



```
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ ps
    PID TTY          TIME CMD
    296 pts/0    00:00:00 bash
   7944 pts/0    00:00:00 ps
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ bash script2.sh
Enter the PID of the process: 296
Details for PID 296:
    PID    PPID S COMMAND         %MEM %CPU
    296     295 S bash             0.1  0.0
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$
```

Task 3

Task Statement:

Create a script that schedules a task to append the current date and time to a log file every minute using cron.

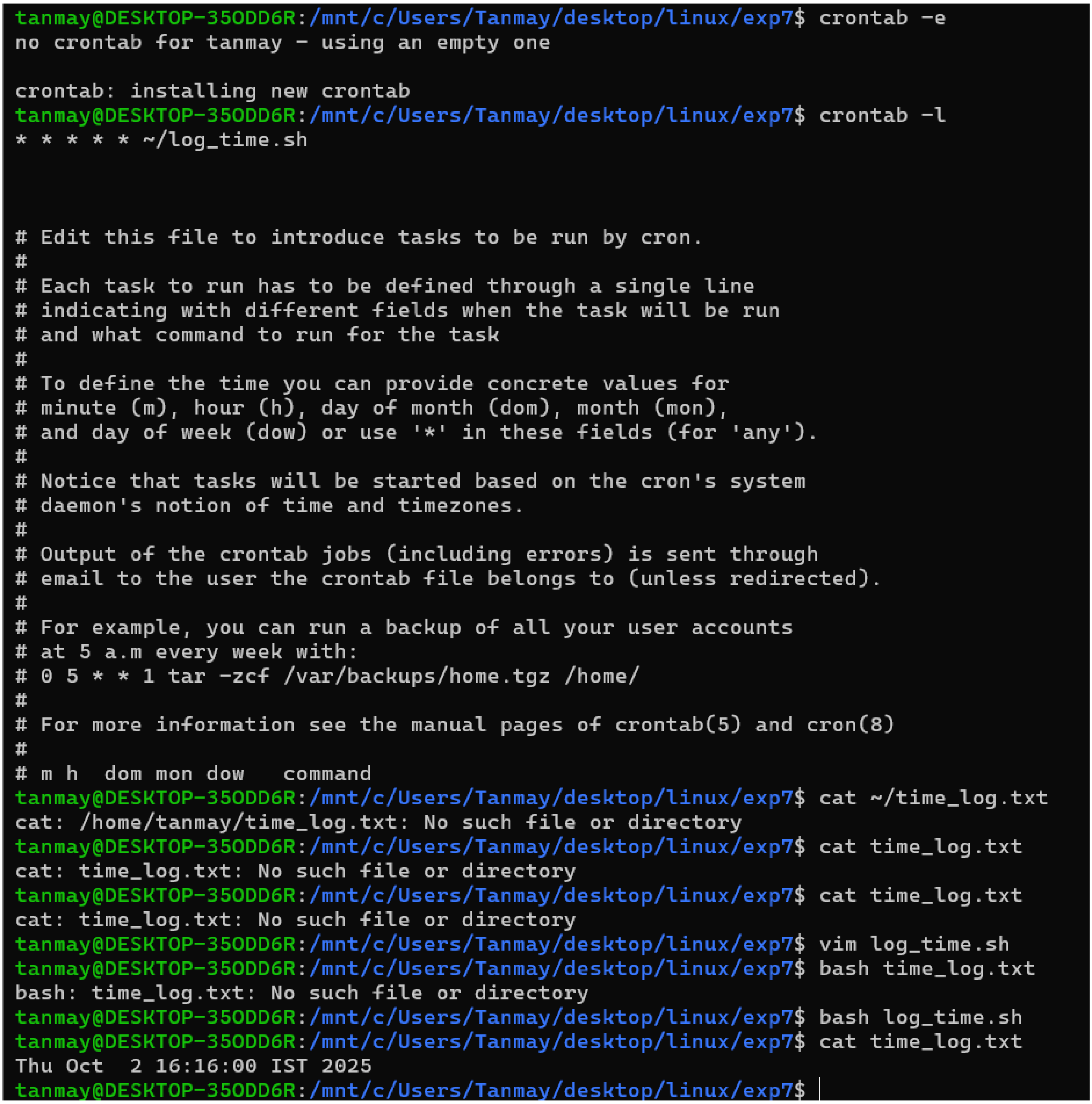
Command(s):

```
#!/bin/bash
echo "$(date)" >> time_log.txt
```

```
crontab -e

* * * * * ~/log_time.sh
```

Output:



```
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ crontab -e
no crontab for tanmay - using an empty one

crontab: installing new crontab
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ crontab -l
* * * * * ~/log_time.sh



# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ cat ~/time_log.txt
cat: /home/tanmay/time_log.txt: No such file or directory
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ cat time_log.txt
cat: time_log.txt: No such file or directory
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ cat time_log.txt
cat: time_log.txt: No such file or directory
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ vim log_time.sh
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ bash time_log.txt
bash: time_log.txt: No such file or directory
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ bash log_time.sh
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ cat time_log.txt
Thu Oct  2 16:16:00 IST 2025
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$
```

Task 4:

Task Statement:

Modify the factorial function to check if input is negative. If yes, display an error message.

Command(s):

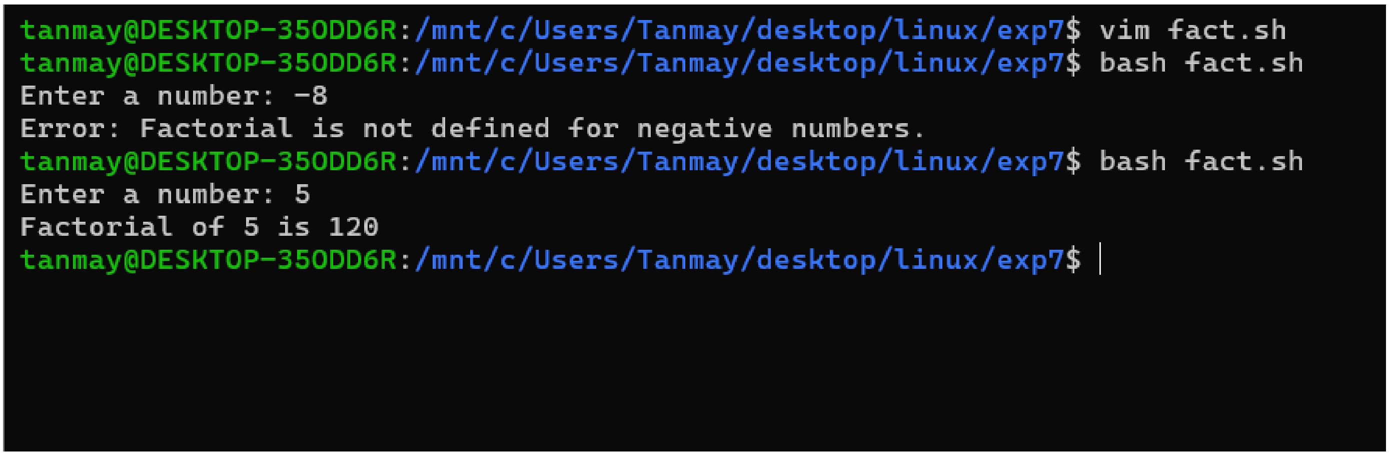
```
#!/bin/bash

factorial() {
    local n=$1
    if [ $n -lt 0 ]; then
        echo "Error: Factorial is not defined for negative numbers."
        return 1
    fi

    local fact=1
    for (( i=1; i<=n; i++ )); do
        fact=$((fact * i))
    done
    echo "Factorial of $n is $fact"
}


read -p "Enter a number: " num
factorial $num
```

Output:



```
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ vim fact.sh
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ bash fact.sh
Enter a number: -8
Error: Factorial is not defined for negative numbers.
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ bash fact.sh
Enter a number: 5
Factorial of 5 is 120
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$
```

Task 5:

Task Statement:

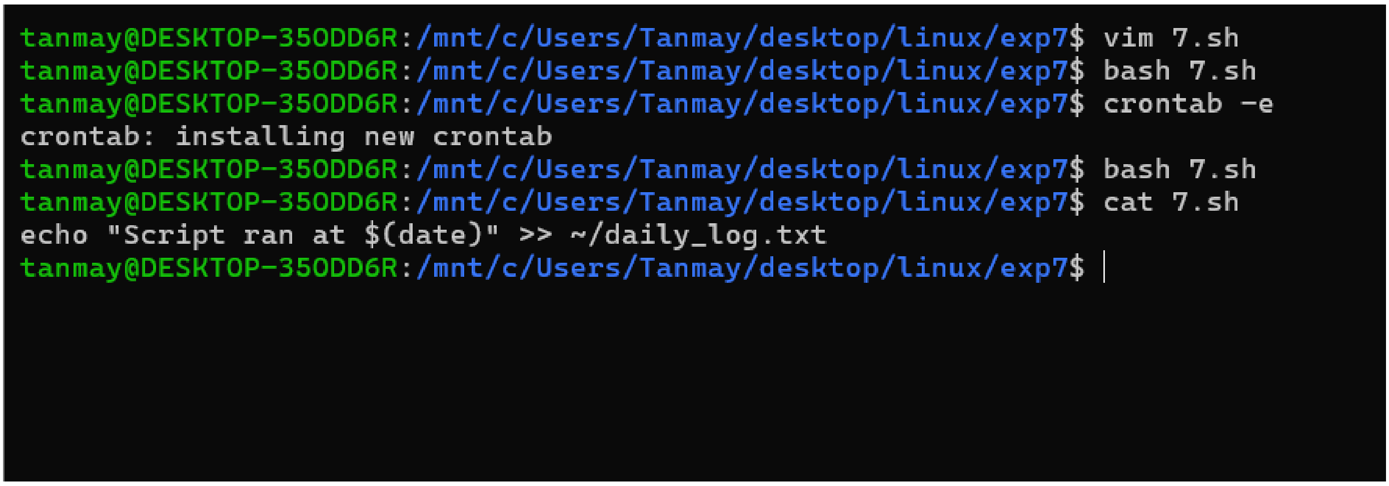
Schedule a script to run every day at 7:00 AM using `cron`.

Command(s):

```
#!/bin/bash
echo "Script ran at $(date)" >> ~/daily_log.txt
```

```
crontab -e
0 7 * * * ~/my_script.sh
```

Output:



```
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ vim 7.sh
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ bash 7.sh
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ crontab -e
crontab: installing new crontab
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ bash 7.sh
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ cat 7.sh
echo "Script ran at $(date)" >> ~/daily_log.txt
tanmay@DESKTOP-35ODD6R:/mnt/c/Users/Tanmay/desktop/linux/exp7$ |
```

Result

- Learned to create and run shell scripts.
- Managed processes using background, foreground, and kill commands.
- Monitored processes with `ps` and `top`.
- Scheduled recurring tasks with `cron` and one-time tasks with `at`.

Challenges Faced & Learning Outcomes

- Challenge 1: Remembering the `crontab` time format. Solved by using online crontab generators and practice.
- Challenge 2: Ensuring `atd` service is running for `at` command. Fixed by starting the service with `systemctl start atd`.

Learning:

- Gained hands-on knowledge of process creation and termination.
- Learned job control and scheduling using `cron` and `at`.

Conclusion

This experiment provided practical experience with shell scripting, process management, and scheduling. These are critical skills for system administrators to automate and control Linux environments effectively.
]