Experiment 7: Shell Programming, Process and Scheduling

Name: Aarush Prasad Roll No.: 590027630 Date: 2025-10-30

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

Exercise 1: Writing a basic shell script

Task Statement:

Create a shell script that prints the current date, time, and the list of logged-in users.

Command(s):

```
#!/bin/bash
echo "Current date and time: $(date)"
echo "Logged in users:"
w
```

Output:

```
aarush07@BlastyPC:~/linux lab$ ls
e1.sh  e2.sh
aarush07@BlastyPC:~/linux lab$ ./e1.sh
Current date and time: Fri Nov 14 18:29:17 UTC 2025
Logged in users:
 18:29:17 up 3 min,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU  WHAT
aarush07 pts/1    -                18:25    3:56   0.04s  0.01s -bash
aarush07@BlastyPC:~/linux lab$
```

Exercise 2: Background and foreground processes

Task Statement:

Run a process in background and bring it to the foreground.

Command(s):

```
sleep 60 &
jobs
fg %1
```

Output:

```
aarush07@BlastyPC:~/linux lab$ ./e2.sh
[1]+  Running                 sleep 60 &
./e2.sh: line 4: fg: no job control
aarush07@BlastyPC:~/linux lab$
```

Exercise 3: Killing a process

Task Statement:

Start a process and terminate it using `kill` .

Command(s):

```
sleep 300 &
ps aux | grep sleep
kill <pid>
```

Output:

```
aarush07@BlastyPC:~/linux lab$ sleep 300 &
[3] 436
aarush07@BlastyPC:~/linux lab$ ps aux | grep sleep
aarush07     424  0.0  0.0   3124  1664 pts/0    S    18:33   0:00 sleep 300
aarush07     433  0.0  0.0   3124  1664 pts/0    S    18:33   0:00 sleep 300
aarush07     436  0.0  0.0   3124  1664 pts/0    S    18:33   0:00 sleep 300
aarush07     438  0.0  0.0   4088  1920 pts/0    S+   18:33   0:00 grep --color=auto sleep
aarush07@BlastyPC:~/linux lab$ kill 438
-bash: kill: (438) - No such process
aarush07@BlastyPC:~/linux lab$
```

Exercise 4: Monitoring processes

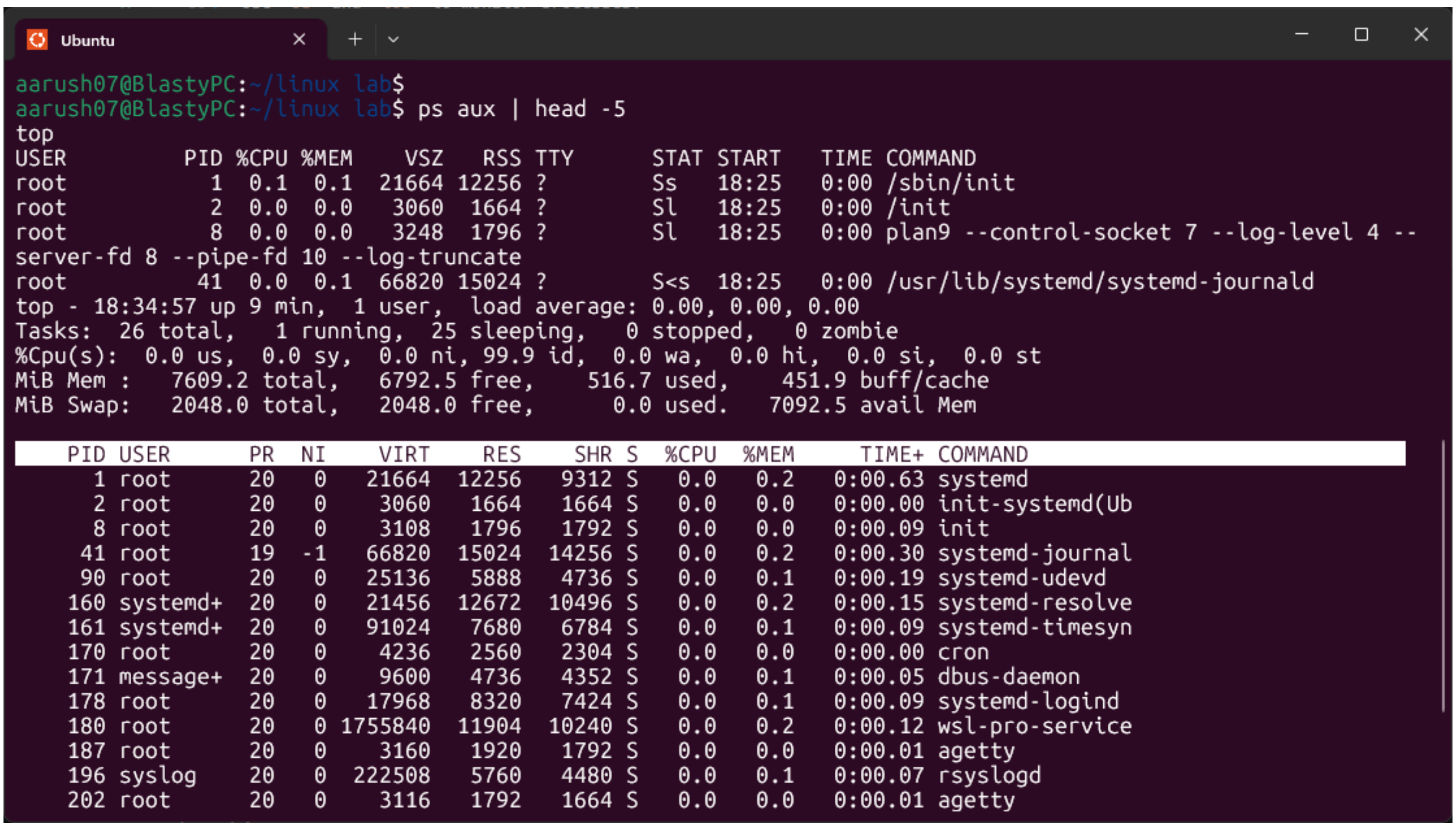
Task Statement:

Use `ps` and `top` to monitor processes.

Command(s):

```
ps aux | head -5
top
```

Output:



Exercise 5: Using `cron` for scheduling

Task Statement:

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

Command(s):

```
crontab -e
# Add the following line
0 7 * * * /home/user/myscript.sh
```

Output:

Exercise 6: Using `at` for one-time scheduling

Task Statement:

Schedule a script to run once at a specified time using `at`.

Command(s):

```
echo "/home/user/myscript.sh" | at 08:30
atq
```

Output:

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.