Experiment 7: Shell Programming, Process and Scheduling

Name:Rahul Roll No.: 590029148 Date: 2025-09-29

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:

```
linuxmint@DESKTOP-KSC4L9L:/mnt/e/liniux$ vim exp7.1.sh
linuxmint@DESKTOP-KSC4L9L:/mnt/e/liniux$ bash exp7.1.sh
Current date and time: Thu Oct 30 18:31:16 UTC 2025
Logged in users:
 18:31:16 up 18 min,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU  WHAT
linuxmin pts/1    -                18:12   18:39   0.10s  0.07s -bash
linuxmint@DESKTOP-KSC4L9L:/mnt/e/liniux$
```

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:

```
linuxmint@DESKTOP-KSC4L9L:/mnt/e/liniux$ sleep 60 &
[1] 592
linuxmint@DESKTOP-KSC4L9L:/mnt/e/liniux$ jobs
[1]+  Running                 sleep 60 &
linuxmint@DESKTOP-KSC4L9L:/mnt/e/liniux$ fg %1
sleep 60
```

Exercise 3: Killing a process

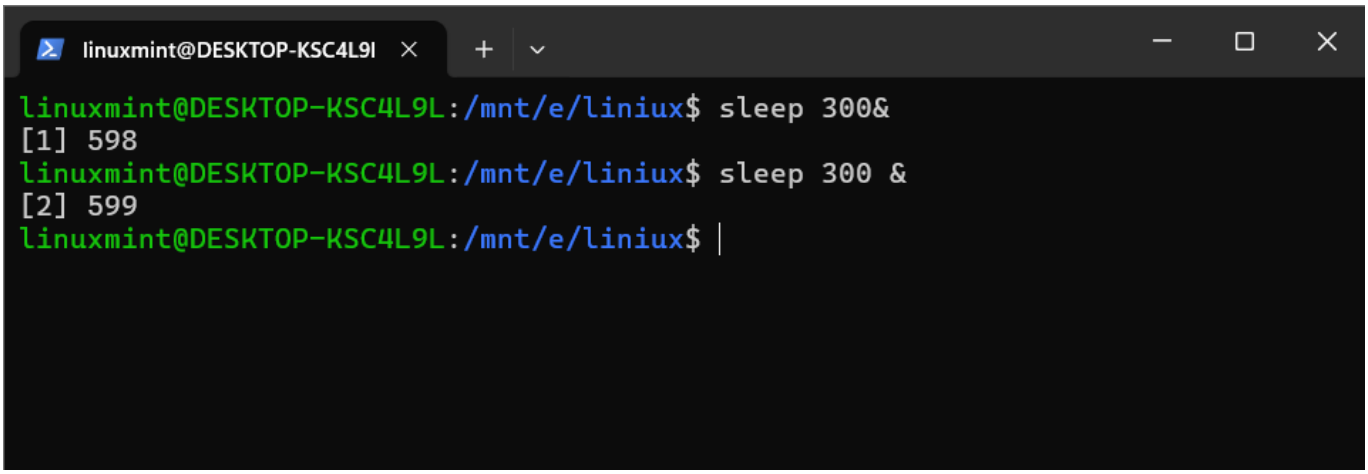
Task Statement:

Start a process and terminate it using `kill`.

Command(s):

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

Output:



Exercise 4: Monitoring processes

Task Statement:

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

Command(s):

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

Output:

```
linuxmint@DESKTOP-KSC4L9I
linuxmint@DESKTOP-KSC4L9L:/mnt/e/liniux$ ps aux | head -5
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.1  21652 12240 ?        Ss   18:13   0:01 /sbin/ini
t
root           2  0.0  0.0   3060  1792 ?        Sl   18:13   0:00 /init
root           8  0.0  0.0   3060  1792 ?        Sl   18:13   0:00 plan9 --c
ontrol-socket 7 --log-level 4 --server-fd 8 --pipe-fd 10 --log-truncate
root          47  0.0  0.2  66816 16560 ?        S<s  18:13   0:00 /usr/lib/
systemd/systemd-journald
linuxmint@DESKTOP-KSC4L9L:/mnt/e/liniux$ top
top - 18:36:35 up 23 min,  1 user,  load average: 0.00, 0.00, 0.00
Tasks:  25 total,   1 running,  24 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni, 99.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.
MiB Mem :   7842.2 total,   7383.8 free,    475.5 used,    132.3 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   7366.7 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+
      1 root      20   0   21652  12240   9296 S   0.0   0.2   0:01.28
      2 root      20   0    3060   1792   1792 S   0.0   0.0   0:00.02
      8 root      20   0    3060   1792   1792 S   0.0   0.0   0:00.00
     47 root      19  -1   66816  16560  15664 S   0.0   0.2   0:00.50
     94 root      20   0   25268   6144   4864 S   0.0   0.1   0:00.32
    118 systemd+  20   0   21456  11904   9984 S   0.0   0.1   0:00.25
    122 systemd+  20   0   91024   7552   6656 S   0.0   0.1   0:00.17
    165 root      20   0    4236   2432   2304 S   0.0   0.0   0:00.02
    166 message+  20   0    9536   4736   4352 S   0.0   0.1   0:00.23
    178 root      20   0   17972   8448   7552 S   0.0   0.1   0:00.20
    181 root      20   0 1756096  12544  10496 S   0.0   0.2   0:00.30
    187 root      20   0    3160   1920   1792 S   0.0   0.0   0:00.02
    200 syslog    20   0  222508   5504   4352 S   0.0   0.1   0:00.19
    204 root      20   0    3116   1792   1664 S   0.0   0.0   0:00.01
    210 root      20   0  107032  22272  13056 S   0.0   0.3   0:00.23
    320 root      20   0    6692   4352   3712 S   0.0   0.1   0:00.03
    360 linuxmi+  20   0   20304  11008   9088 S   0.0   0.1   0:00.38
    361 linuxmi+  20   0   21152   3516   1792 S   0.0   0.0   0:00.00
    403 linuxmi+  20   0    6072   4992   3456 S   0.0   0.1   0:00.07
    452 root      20   0    3068    896    896 S   0.0   0.0   0:00.00
    453 root      20   0    3084   1152   1024 S   0.0   0.0   0:00.57
    454 linuxmi+  20   0    6072   4992   3456 S   0.0   0.1   0:00.22
    598 linuxmi+  20   0    3124   1664   1664 S   0.0   0.0   0:00.00
    599 linuxmi+  20   0    3124   1664   1664 S   0.0   0.0   0:00.00
    605 linuxmi+  20   0    9296   5504   3328 R   0.0   0.1   0:00.01
```

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

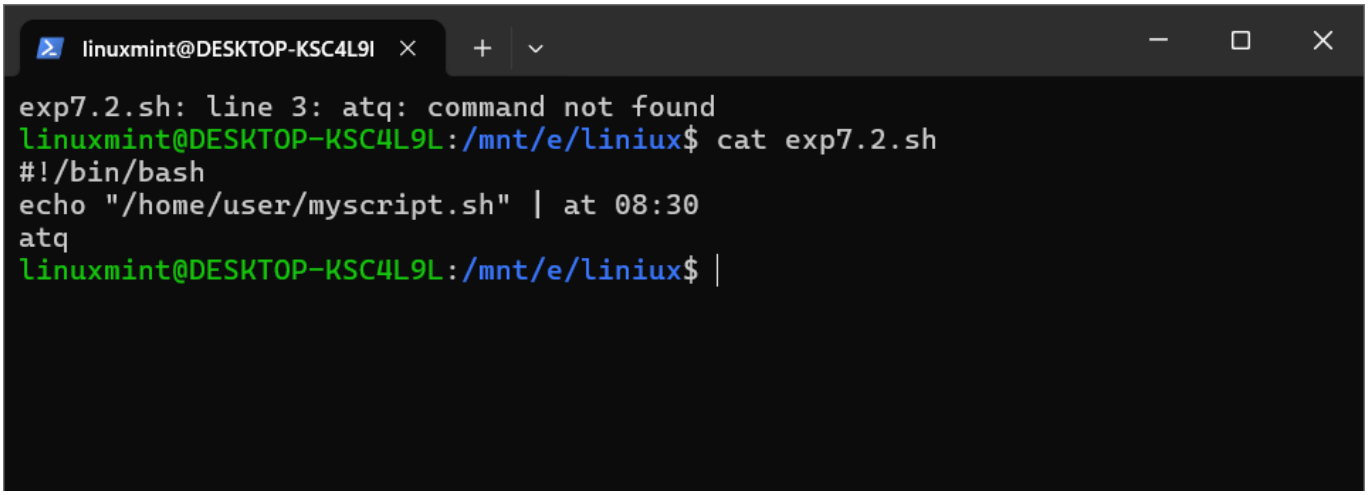
Task Statement:

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.