

Experiment 7: Shell Programming, Process and Scheduling

Name: Biswabandya Mohanty Roll No.: 590029274 Date: 2025-11-1

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:

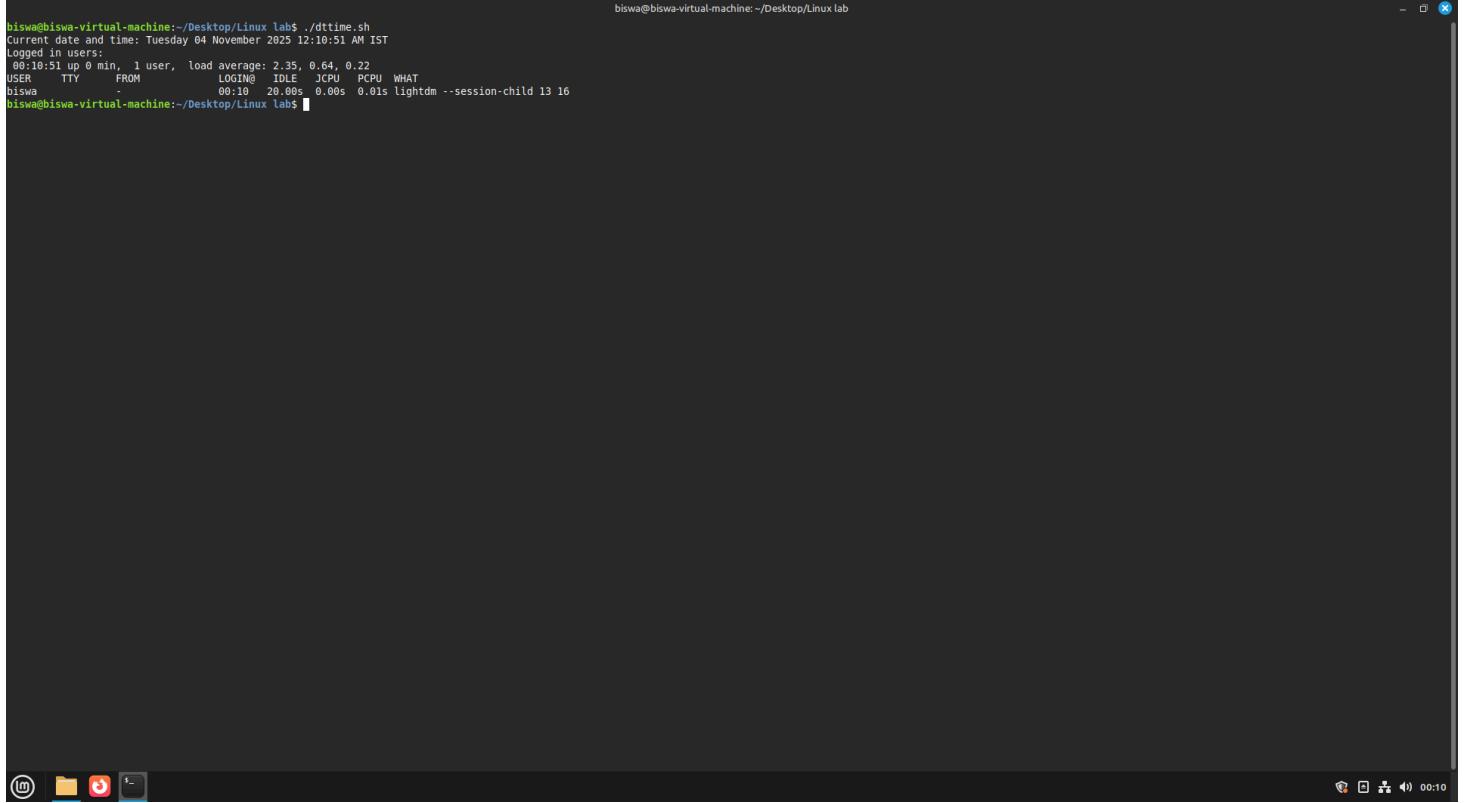
Create a shell script that prints the current date, time, and the list of logged-in users.

Command(s):

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

W

Output:



biswa@biswa-virtual-machine:~/Desktop/Linux lab\$./dttime.sh
Current date and time: Tuesday 04 November 2023 12:10:51 AM IST
Logged in users:
 00:10:51 up 0 min, 1 user, load average: 2.35, 0.64, 0.22
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
biswa - 00:10 20.00s 0.00s 0.01s lightdm --session-child 13 16
biswa@biswa-virtual-machine:~/Desktop/Linux lab\$

Exercise 2: Background and foreground processes

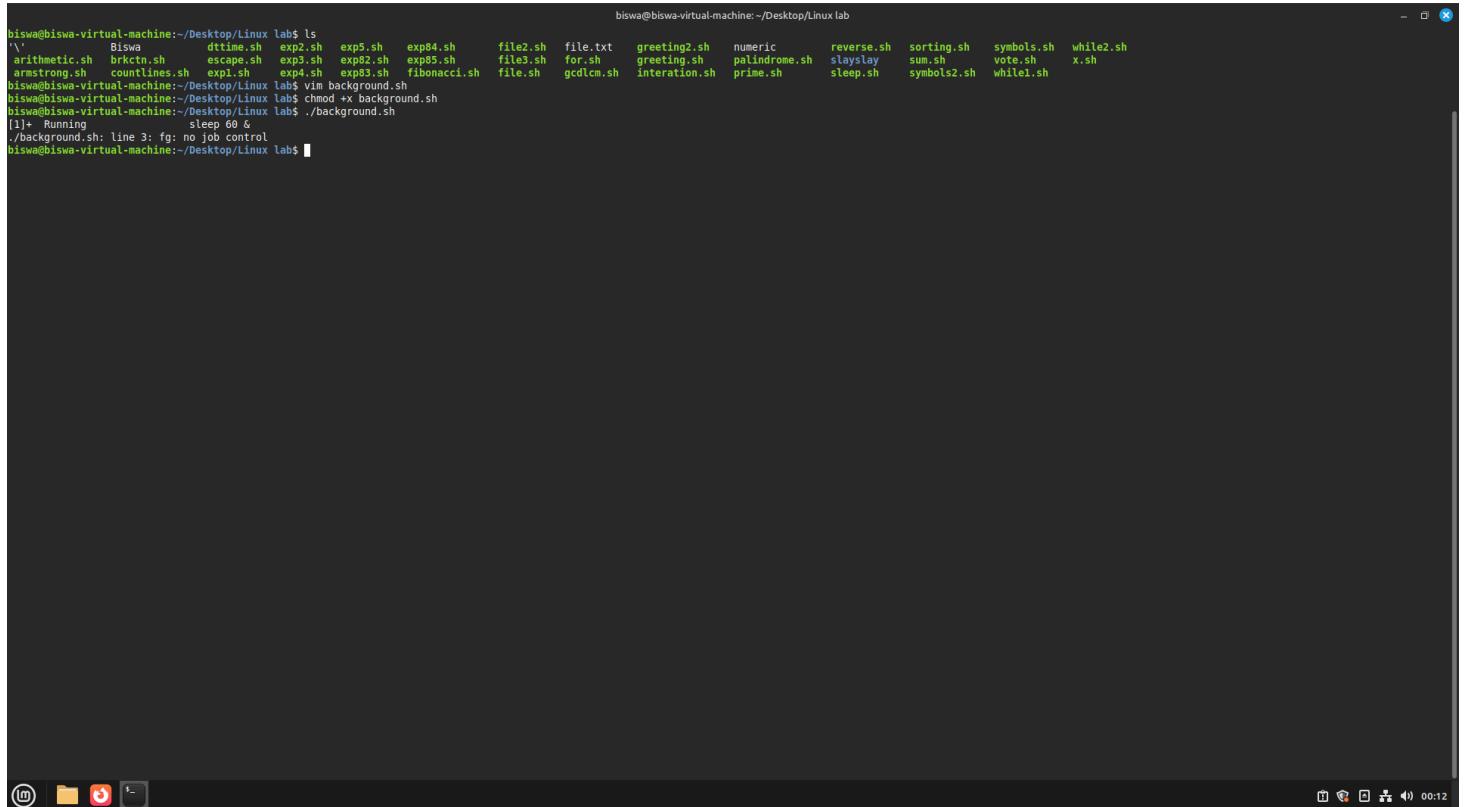
Task Statement:

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

Command(s):

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

Output:



```
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ls
'.' Biswa dttime.sh exp2.sh exp5.sh exp84.sh file2.sh file.txt greeting2.sh numeric reverse.sh sorting.sh symbols.sh while2.sh
'..' brkctn.sh escape.sh exp3.sh exp82.sh exp85.sh file3.sh for.sh greeting.sh palindrome.sh slayslay sum.sh vote.sh x.sh
armstrong.sh countlines.sh expl.sh exp4.sh exp83.sh fibonacci.sh file.sh gcdlcm.sh interation.sh prime.sh sleep.sh symbolsz.sh white1.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ vim background.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ chmod +x background.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ./background.sh
[1]+  Running                  sleep 60 &
./background.sh: line 3: fg: no job control
biswa@biswa-virtual-machine:~/Desktop/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:

```
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ls
'`' background.sh countlines.sh expl.sh exp4.sh exp83.sh fibonacci.sh file.sh gcdlcm.sh interation.sh palindrome.sh slayslay sum.sh vote.sh x.sh
arithmetic.sh Biswa dttime.sh exp2.sh exp5.sh exp84.sh file2.sh file.txt gcdlcm.sh interation.sh palindrome.sh slayslay sum.sh vote.sh x.sh
armstrong.sh brkctn.sh escape.sh exp3.sh exp82.sh exp85.sh file3.sh for.sh greeting.sh numeric kill.sh prime.sh sleep.sh symbols2.sh while1.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ vim kill.sh
biswa@biswa-virtual-machine:~/Desktop/Linux lab$ ./kill.sh
biswa 2419 0.0 0.0 6288 1920 ? S 00:14 0:00 sleep 300
biswa 2550 0.0 0.0 6288 1920 pts/1 S+ 00:17 0:00 sleep 300
biswa 2600 0.0 0.0 9144 2176 pts/1 S+ 00:17 0:00 grep sleep
./kill.sh: line 4: syntax error near unexpected token `newline'
./kill.sh: line 4: kill <pid>
biswa@biswa-virtual-machine:~/Desktop/Linux lab$
```

Exercise 4: Monitoring processes

Task Statement:

Use ps and top to monitor processes.

Command(s):

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

Output:

```
biswa@biswa-virtual-machine:~/Desktop/Linux lab
top - 00:21:46 up 11 min, 1 user, load average: 0.49, 0.33, 0.22
Tasks: 287 total, 1 running, 286 sleeping, 0 stopped, 0 zombie
%CPU(s): 2.2 us, 0.5 sy, 0.0 ni, 97.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
Mem: 1920.0 total, 193.8 free, 1175.6 used, 766.9 buff/cache
Swap: 6046.0 total, 0.0 free, 0.0 used, 744.4 avail Mem

PID USER      PR  NI  VIRT   RES   SHR S %CPU %MEM TIME+ COMMAND
1748 biswa    20   0 388048 111232 176684 S  3.0 15.8 1:02.01 cinnamon
1801 biswa    20   0 592884 186000 108084 S  1.0  9.5 0:11.94 nautilus
2269 biswa    20   0 566048 59456 36096 S  0.7  3.0 0:01.67 mintreport-tray
1883 biswa    20   0 246488 36776 29608 S  0.3  1.9 0:00.92 vntoolsld
2851 biswa    20   0 1061972 80556 48188 S  0.3  4.1 0:01.23 nemo
2524 biswa    20   0 545208 46040 33664 S  0.3  2.3 0:00.89 gnome-terminal-
2636 biswa    20   0 14568 5760 3584 R  0.3  0.3 0:00.12 top
1 root      20   0 22380 13540 9444 S  0.0  0.7 0:01.66 systemd
2 root      20   0       0  0       0 S  0.0  0.0 0:00.01 kthread
3 root      20   0       0  0       0 S  0.0  0.0 0:00.00 pool.workqueue_release
4 root      0 -20  0       0  0       0 I  0.0  0.0 0:00.00 kworker/R-rcu_g
5 root      0 -20  0       0  0       0 I  0.0  0.0 0:00.00 kworker/R-rcu_p
6 root      0 -20  0       0  0       0 I  0.0  0.0 0:00.00 kworker/R-slub
7 root      0 -20  0       0  0       0 I  0.0  0.0 0:00.00 kworker/R-journal
9 root      20   0       0  0       0 I  0.0  0.0 0:00.02 kworker/0:1:cgroup destroy
10 root     0 -20  0       0  0       0 I  0.0  0.0 0:00.00 kworker/0:0h-events_highpri
11 root     20   0       0  0       0 I  0.0  0.0 0:00.00 kworker/u256:0-ext4-rsv-conversion
12 root     0 -20  0       0  0       0 I  0.0  0.0 0:00.00 kworker/mm_pe
13 root     20   0       0  0       0 I  0.0  0.0 0:00.00 rCU_tasks_kthread
14 root     20   0       0  0       0 I  0.0  0.0 0:00.00 rCU_tasks_rude_kthread
15 root     20   0       0  0       0 I  0.0  0.0 0:00.00 rCU_tasks_trace_kthread
16 root     20   0       0  0       0 S  0.0  0.0 0:00.01 ksoftirqd/0
17 root     20   0       0  0       0 I  0.0  0.0 0:00.07 rCU_prempt
18 root     rt   0       0  0       0 S  0.0  0.0 0:00.01 migration/0
19 root     -51  0       0  0       0 S  0.0  0.0 0:00.00 ksoftirqd/0
20 root     20   0       0  0       0 S  0.0  0.0 0:00.00 cpuhp/0
21 root     20   0       0  0       0 S  0.0  0.0 0:00.00 cpuhp/1
22 root     -51  0       0  0       0 S  0.0  0.0 0:00.00 idle_inject/1
23 root     rt   0       0  0       0 S  0.0  0.0 0:00.18 migration/1
24 root     20   0       0  0       0 S  0.0  0.0 0:00.01 ksoftirqd/1
25 root     20   0       0  0       0 I  0.0  0.0 0:00.00 kworker/1:0-events
26 root     0 -20  0       0  0       0 I  0.0  0.0 0:00.01 kworker/1:0h-events_highpri
29 root     20   0       0  0       0 S  0.0  0.0 0:00.01 kdevtmpfs
30 root     0 -20  0       0  0       0 I  0.0  0.0 0:00.00 kworker/R-inet_
31 root     20   0       0  0       0 S  0.0  0.0 0:00.00 kauidtd
34 root     20   0       0  0       0 S  0.0  0.0 0:00.00 khungtaskd
35 root     20   0       0  0       0 S  0.0  0.0 0:00.00 kworker/0:0-power
36 root     20   0       0  0       0 I  0.0  0.0 0:00.32 kworker/u256:1-events_power_efficient
37 root     0 -20  0       0  0       0 I  0.0  0.0 0:00.00 kworker/R-write
38 root     20   0       0  0       0 S  0.0  0.0 0:00.01 kcompactd0
39 root     25  5       0  0       0 S  0.0  0.0 0:00.00 ksmd
42 root     39 19      0  0       0 S  0.0  0.0 0:00.00 khugepaged
43 root     0 -20  0       0  0       0 I  0.0  0.0 0:00.00 kworker/R-kinte
44 root     0 -20  0       0  0       0 I  0.0  0.0 0:00.00 kworker/R-kblob
45 root     0 -20  0       0  0       0 I  0.0  0.0 0:00.00 kworker/R-blkg
46 root     -51  0       0  0       0 S  0.0  0.0 0:00.00 irq/9-apci
47 root     0 -20  0       0  0       0 I  0.0  0.0 0:00.00 kworker/R-tpm_d
48 root     0 -20  0       0  0       0 I  0.0  0.0 0:00.00 kworker/R-ata_s
49 root     0 -20  0       0  0       0 I  0.0  0.0 0:00.00 kworker/R-ata_t
50 root     0 -20  0       0  0       0 I  0.0  0.0 0:00.00 kworker/R-md_b1
51 root     0 -20  0       0  0       0 I  0.0  0.0 0:00.00 kworker/R-edac-
```

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
0 7 * * * /home/retr0/myscript.sh
```

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
```

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.