Experiment 7: Shell Programming, Process and Scheduling

Name: Tanishq bansal Roll No.: 590029349 Date: 2025-10-29

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

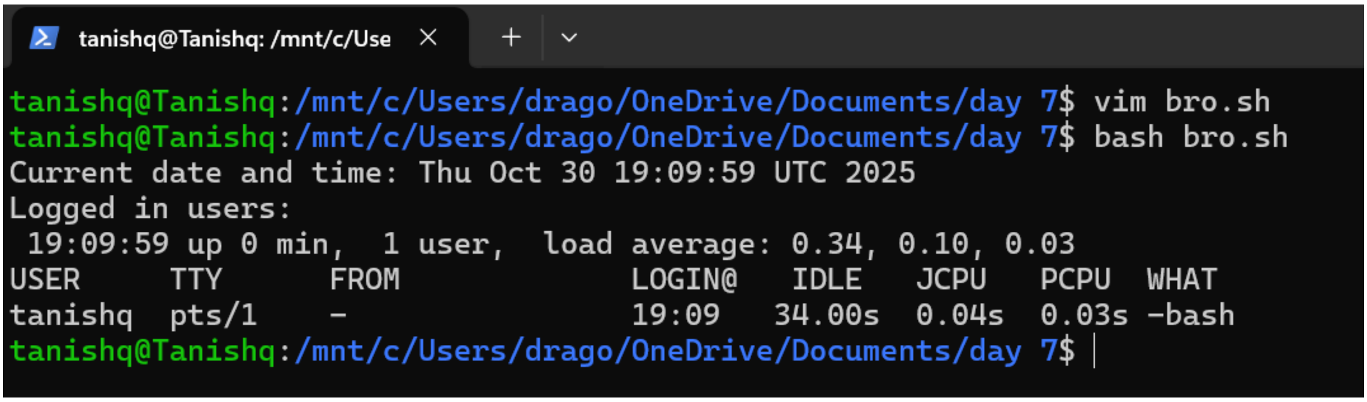
Task Statement:

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:



```
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 7$ vim bro.sh
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 7$ bash bro.sh
Current date and time: Thu Oct 30 19:09:59 UTC 2025
Logged in users:
 19:09:59 up 0 min,  1 user,  load average: 0.34, 0.10, 0.03
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU  WHAT
tanishq  pts/1    -                19:09   34.00s  0.04s  0.03s -bash
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 7$
```

Exercise 2: Background and foreground processes

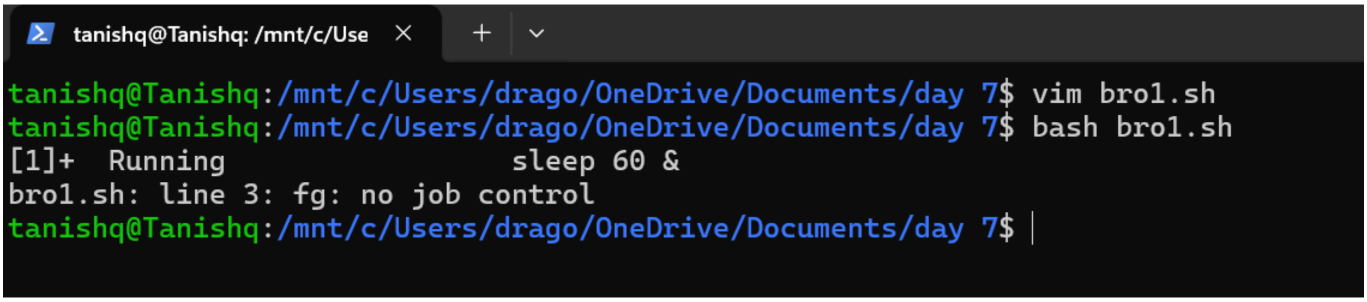
Task Statement:

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

Command(s):

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

Output:



```
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 7$ vim bro1.sh
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 7$ bash bro1.sh
[1]+  Running                 sleep 60 &
bro1.sh: line 3: fg: no job control
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 7$
```

Exercise 3: Killing a process

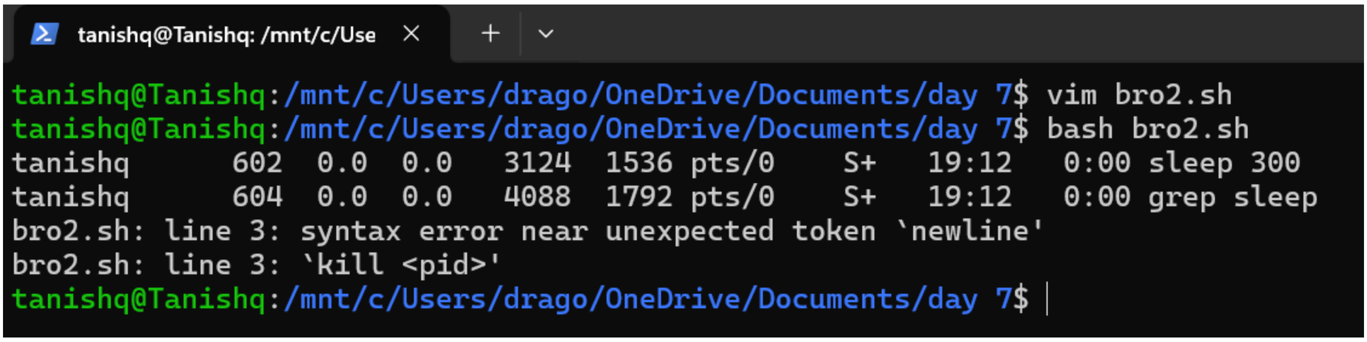
Task Statement:

Start a process and terminate it using `kill`.

Command(s):

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

Output:



```
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 7$ vim bro2.sh
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 7$ bash bro2.sh
tanishq      602  0.0  0.0   3124  1536 pts/0    S+   19:12   0:00 sleep 300
tanishq      604  0.0  0.0   4088  1792 pts/0    S+   19:12   0:00 grep sleep
bro2.sh: line 3: syntax error near unexpected token `newline'
bro2.sh: line 3: `kill <pid>'
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 7$
```

Exercise 4: Monitoring processes

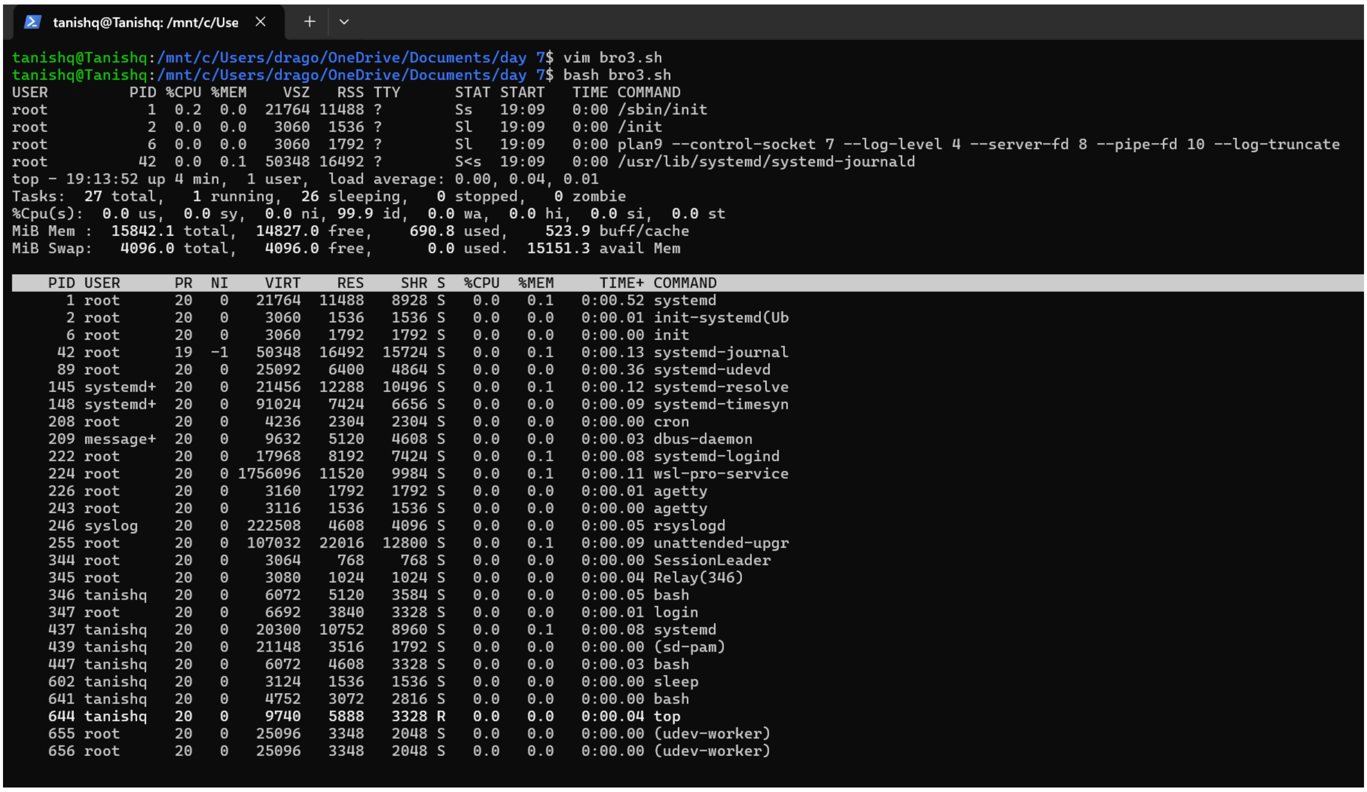
Task Statement:

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

Command(s):

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

Output:



```
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 7$ vim bro3.sh
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 7$ bash bro3.sh
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.2  0.0  21764 11488 ?        Ss   19:09   0:00 /sbin/init
root           2  0.0  0.0   3060  1536 ?        Sl   19:09   0:00 /init
root           6  0.0  0.0   3060  1792 ?        Sl   19:09   0:00 plan9 --control-socket 7 --log-level 4 --server-fd 8 --pipe-fd 10 --log-truncate
root          42  0.0  0.1  50348 16492 ?        S<s  19:09   0:00 /usr/lib/systemd/systemd-journald
top - 19:13:52 up 4 min,  1 user,  load average: 0.00, 0.04, 0.01
Tasks:  27 total,   1 running,  26 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,  0.0 st
MiB Mem :  15842.1 total,  14827.0 free,    690.8 used,    523.9 buff/cache
MiB Swap:   4096.0 total,   4096.0 free,      0.0 used.  15151.3 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
      1 root      20   0   21764  11488   8928 S   0.0   0.1   0:00.52 systemd
      2 root      20   0    3060   1536   1536 S   0.0   0.0   0:00.01 init-systemd(Ub
      6 root      20   0    3060   1792   1792 S   0.0   0.0   0:00.00 init
     42 root      19  -1   50348  16492  15724 S   0.0   0.1   0:00.13 systemd-journal
     89 root      20   0   25092   6400   4864 S   0.0   0.0   0:00.36 systemd-udevd
    145 systemd+  20   0   21456  12288  10496 S   0.0   0.1   0:00.12 systemd-resolve
    148 systemd+  20   0   91024   7424   6656 S   0.0   0.0   0:00.09 systemd-timesyn
    208 root      20   0    4236   2304   2304 S   0.0   0.0   0:00.00 cron
    209 message+  20   0    9632   5120   4608 S   0.0   0.0   0:00.03 dbus-daemon
    222 root      20   0   17968   8192   7424 S   0.0   0.1   0:00.08 systemd-logind
    224 root      20   0 1756096  11520   9984 S   0.0   0.1   0:00.11 wsl-pro-service
    226 root      20   0    3160   1792   1792 S   0.0   0.0   0:00.01 agetty
    243 root      20   0    3116   1536   1536 S   0.0   0.0   0:00.00 agetty
    246 syslog    20   0  222508   4608   4096 S   0.0   0.0   0:00.05 rsyslogd
    255 root      20   0  107032  22016  12800 S   0.0   0.1   0:00.09 unattended-upgr
    344 root      20   0    3064    768    768 S   0.0   0.0   0:00.00 SessionLeader
    345 root      20   0    3080   1024   1024 S   0.0   0.0   0:00.04 Relay(346)
    346 tanishq   20   0    6072   5120   3584 S   0.0   0.0   0:00.05 bash
    347 root      20   0    6692   3840   3328 S   0.0   0.0   0:00.01 login
    437 tanishq   20   0   20300  10752   8960 S   0.0   0.1   0:00.08 systemd
    439 tanishq   20   0   21148   3516   1792 S   0.0   0.0   0:00.00 (sd-pam)
    447 tanishq   20   0    6072   4608   3328 S   0.0   0.0   0:00.03 bash
    602 tanishq   20   0    3124   1536   1536 S   0.0   0.0   0:00.00 sleep
    641 tanishq   20   0    4752   3072   2816 S   0.0   0.0   0:00.00 bash
    644 tanishq   20   0    9740   5888   3328 R   0.0   0.0   0:00.04 top
    655 root      20   0   25096   3348   2048 S   0.0   0.0   0:00.00 (udev-worker)
    656 root      20   0   25096   3348   2048 S   0.0   0.0   0:00.00 (udev-worker)
```

Exercise 5: Using `cron` for scheduling

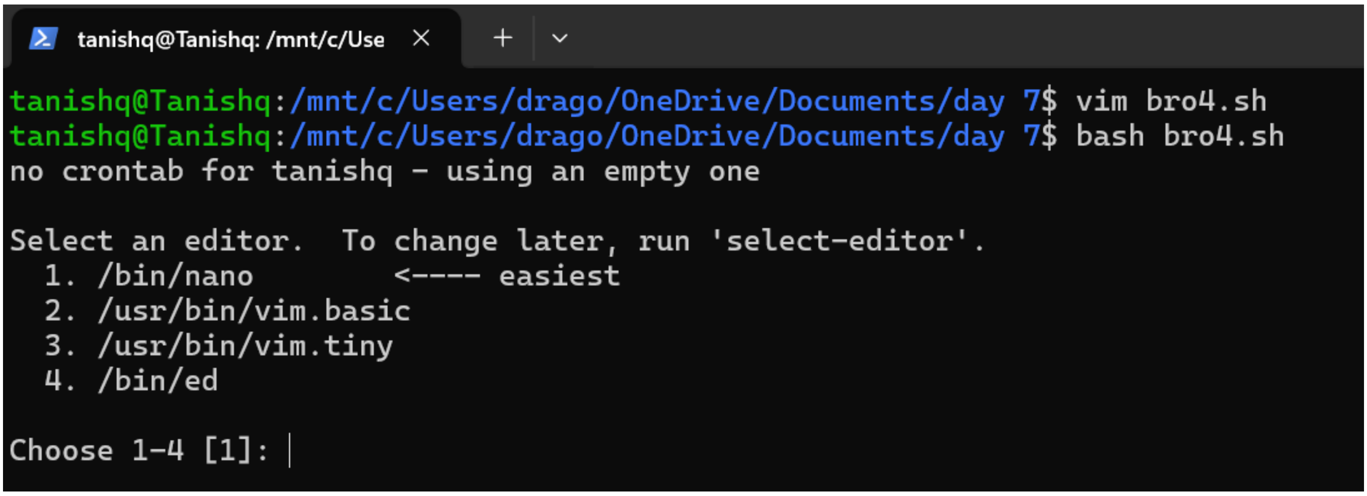
Task Statement:

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:



```
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 7$ vim bro4.sh
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 7$ bash bro4.sh
no crontab for tanishq - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.basic
  3. /usr/bin/vim.tiny
  4. /bin/ed

Choose 1-4 [1]:
```

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

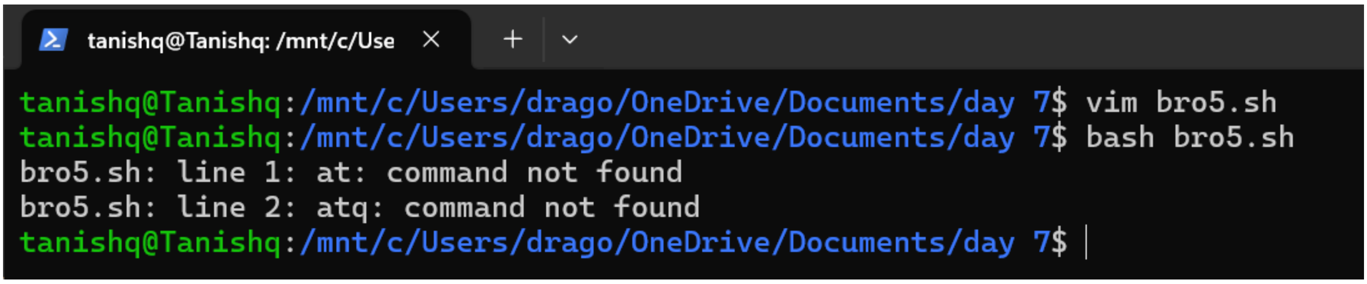
Task Statement:

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

Command(s):

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

Output:



```
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 7$ vim bro5.sh
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 7$ bash bro5.sh
bro5.sh: line 1: at: command not found
bro5.sh: line 2: atq: command not found
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 7$
```

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.