[590023785]Exp7_Scrip\[590023785]Exp7_Scrip.md

Experiment 7: Shell Programming, Process and Scheduling

Name: Gayatri Bhatt Roll No.: 590023785 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

In Linux, every running application is represented as a unique process, distinguished by its process ID (PID). Shell scripts enable the automation of process creation, management, and termination. Essential process control utilities, including `ps`, `top`, `kill`, `jobs`, `bg`, and `fg`, help users observe and influence program execution. Automation tools like `cron` (for periodic tasks) and `at` (for single-scheduled tasks) allow processes to be executed at set times without manual intervention. Developing proficiency in scripting alongside process scheduling is an essential component of effective system administration.

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

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ vim 1ex1.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ bash 1ex1.sh
Current date and time: Wed Sep 24 06:34:34 UTC 2025
Logged in users:
 06:34:34 up 2 min,  1 user,  load average: 0.09, 0.05, 0.01
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU  WHAT
gayatri1 pts/1    -                06:32    1:57   0.03s  0.02s -bash
```

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

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ vim 1ex2.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ bash 1ex2.sh
[1]+  Running                 sleep 60 &
1ex2.sh: line 4: fg: no job control
```

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

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ vim 1ex3.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ bash 1ex3.sh
gayatri+     483  0.0  0.0   3124  1664 pts/0    S    06:35   0:00 sleep 60
gayatri+     487  0.0  0.0   3124  1664 pts/0    S+   06:36   0:00 sleep 300
gayatri+     489  0.0  0.0   4088  1920 pts/0    S+   06:36   0:00 grep sleep
1ex3.sh: line 4: syntax error near unexpected token `newline'
1ex3.sh: line 4: `kill <pid>'
```

Exercise 4: Monitoring processes

Task Statement

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

Command(s)

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

```
top
```

Output

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ vim 1ex4.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ bash 1ex4.sh
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.3  0.1  21560 12156 ?        Ss   06:32   0:01 /sbin/init
root           2  0.0  0.0   3060  1664 ?        Sl   06:32   0:00 /init
root           6  0.0  0.0   3060  1792 ?        Sl   06:32   0:00 plan9 --control-socket 7 --log-level 4 --server-fd 8 --pipe-fd 10 --log-truncate
root          42  0.1  0.1  66824 15068 ?        S<s  06:32   0:00 /usr/lib/systemd/systemd-journald
top - 06:38:12 up 5 min,  1 user,  load average: 0.00, 0.02, 0.00
Tasks:  25 total,   1 running,  24 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.7 sy,  0.0 ni, 99.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.
MiB Mem :   7560.2 total,   6728.0 free,    498.2 used,    483.0 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   7062.0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+
    497 gayatri+  20   0    9292   5504   3328 R   5.9   0.1   0:00.03
      1 root      20   0   21560  12156   9212 S   0.0   0.2   0:01.06
      2 root      20   0    3060   1664   1664 S   0.0   0.0   0:00.01
      6 root      20   0    3060   1792   1792 S   0.0   0.0   0:00.00
     42 root      19  -1   66824  15068  14300 S   0.0   0.2   0:00.62
     92 root      20   0   25136   6144   4864 S   0.0   0.1   0:00.25
    147 systemd+  20   0   21456  12544  10368 S   0.0   0.2   0:00.21
    148 systemd+  20   0   91024   7680   6784 S   0.0   0.1   0:00.15
    166 root      20   0    4236   2432   2304 S   0.0   0.0   0:00.01
    167 message+  20   0    9624   4736   4352 S   0.0   0.1   0:00.09
    178 root      20   0   17964   8192   7296 S   0.0   0.1   0:00.13
    182 root      20   0 1755840  11904  10240 S   0.0   0.2   0:00.20
    185 syslog    20   0  222508   5504   4352 S   0.0   0.1   0:00.09
    203 root      20   0    3160   1920   1792 S   0.0   0.0   0:00.02
    209 root      20   0    3116   1792   1664 S   0.0   0.0   0:00.01
    215 root      20   0  107032  21880  12800 S   0.0   0.3   0:00.25
    298 root      20   0    3064    896    896 S   0.0   0.0   0:00.00
    299 root      20   0    3080   1024   1024 S   0.0   0.0   0:00.07
    304 gayatri+  20   0    6068   4864   3456 S   0.0   0.1   0:00.08
    306 root      20   0    6692   4224   3584 S   0.0   0.1   0:00.01
    400 gayatri+  20   0   20292  10752   8960 S   0.0   0.1   0:00.14
    402 gayatri+  20   0   21156   3516   1792 S   0.0   0.0   0:00.00
    444 gayatri+  20   0    6056   5120   3456 S   0.0   0.1   0:00.02
```

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

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ vim 1ex5.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1$ bash 1ex5.sh
no crontab for gayatri10 - 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]: 2
crontab: installing new crontab
1ex5.sh: line 4: 0: command not found
```

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

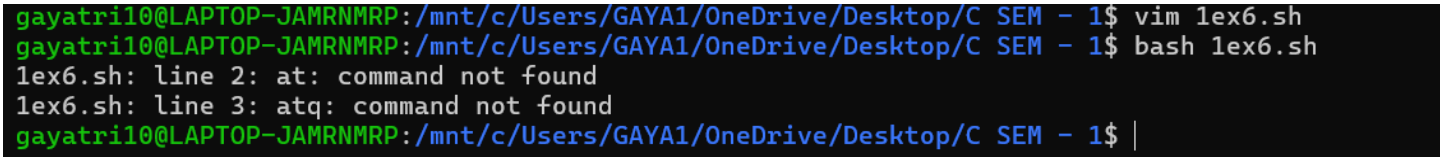
Task Statement

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

Command(s)

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

Output



Task 1

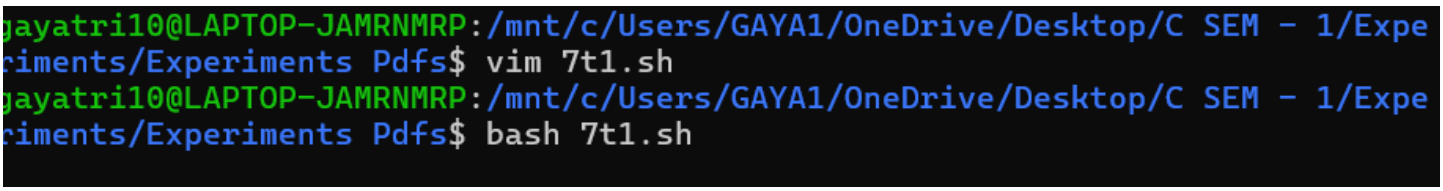
Task Statement

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



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

```
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Experiments/Experiments Pdfs$ vim 7t2.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Experiments/Experiments Pdfs$ bash 7t2.sh
Enter the PID of the process: #!/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
Details for PID #!/bin/bash:
error: process ID list syntax error

Usage:
 ps [options]

 Try 'ps --help <simple|list|output|threads|misc|all>'
  or 'ps --help <s|l|o|t|m|a>'
 for additional help text.

For more details see ps(1).
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Experiments/Experiments Pdfs$
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Experiments/Experiments Pdfs$ read -p "Enter the PID of the process: " pid
Enter the PID of the process: gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Experiments/Experiments Pdfs$ read -p "Enter the PID o
f the process: " pid  ps
    PID TTY          TIME CMD
    303 pts/0    00:00:00 bash
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Experiments/Experiments Pdfs$
```

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

```
# 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
~
~
~
```

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

```
riments/Experiments Pdfs$ vim 7t4.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Expe
riments/Experiments Pdfs$ bash 7t4.sh
Enter a number: 2
Factorial of 2 is 2
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Expe
riments/Experiments Pdfs$ bash 7t4.sh
Enter a number: 8
Factorial of 8 is 40320
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Expe
riments/Experiments Pdfs$
```

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

```
riments/Experiments Pdfs$ vim 7t5.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Expe
riments/Experiments Pdfs$ bash 7t5.sh
gayatri10@LAPTOP-JAMRNMRP:/mnt/c/Users/GAYA1/OneDrive/Desktop/C SEM - 1/Expe
riments/Experiments Pdfs$
```

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.