

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:

```
Ubuntu x + v
aarush07@BlastyPC:~/linux lab$
aarush07@BlastyPC:~/linux lab$ ps aux | head -5
top
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.1  0.1 21664 12256 ?        Ss   18:25   0:00 /sbin/init
root         2  0.0  0.0  3060  1664 ?        Sl   18:25   0:00 /init
root         8  0.0  0.0  3248  1796 ?        Sl   18:25   0:00 plan9 --control-socket 7 --log-level 4 --
server-fd 8 --pipe-fd 10 --log-truncate
root        41  0.0  0.1 66820 15024 ?        S<s  18:25   0:00 /usr/lib/systemd/systemd-journald
top - 18:34:57 up 9 min,  1 user,  load average: 0.00, 0.00, 0.00
Tasks: 26 total,  1 running, 25 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 : 7609.2 total, 6792.5 free,   516.7 used,   451.9 buff/cache
MiB Swap: 2048.0 total, 2048.0 free,    0.0 used. 7092.5 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1	root	20	0	21664	12256	9312	S	0.0	0.2	0:00.63	systemd
2	root	20	0	3060	1664	1664	S	0.0	0.0	0:00.00	init-systemd(Ub
8	root	20	0	3108	1796	1792	S	0.0	0.0	0:00.09	init
41	root	19	-1	66820	15024	14256	S	0.0	0.2	0:00.30	systemd-journal
90	root	20	0	25136	5888	4736	S	0.0	0.1	0:00.19	systemd-udev
160	systemd+	20	0	21456	12672	10496	S	0.0	0.2	0:00.15	systemd-resolve
161	systemd+	20	0	91024	7680	6784	S	0.0	0.1	0:00.09	systemd-timesyn
170	root	20	0	4236	2560	2304	S	0.0	0.0	0:00.00	cron
171	message+	20	0	9600	4736	4352	S	0.0	0.1	0:00.05	dbus-daemon
178	root	20	0	17968	8320	7424	S	0.0	0.1	0:00.09	systemd-logind
180	root	20	0	1755840	11904	10240	S	0.0	0.2	0:00.12	wsl-pro-service
187	root	20	0	3160	1920	1792	S	0.0	0.0	0:00.01	agetty
196	syslog	20	0	222508	5760	4480	S	0.0	0.1	0:00.07	rsyslogd
202	root	20	0	3116	1792	1664	S	0.0	0.0	0:00.01	agetty

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.