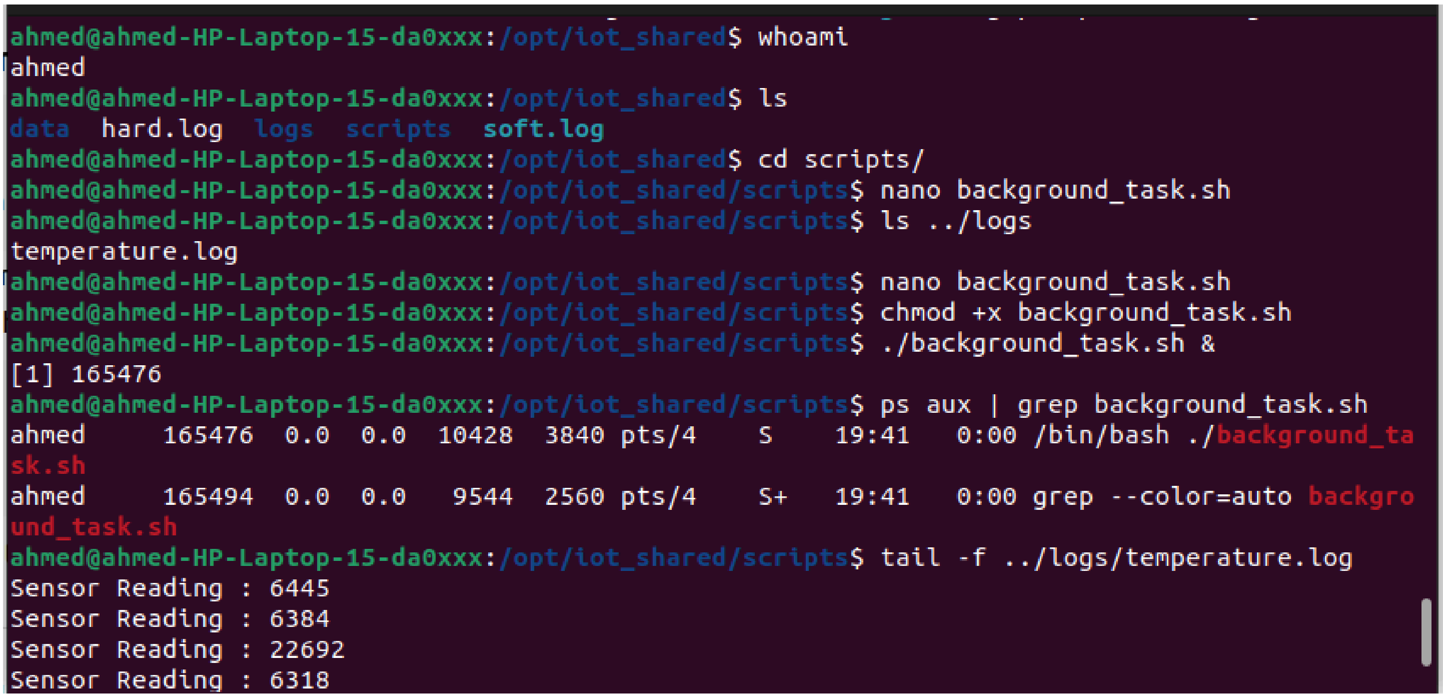
Name : Ahmed Khaled Abdelmaksod

Group : IOT_701_O

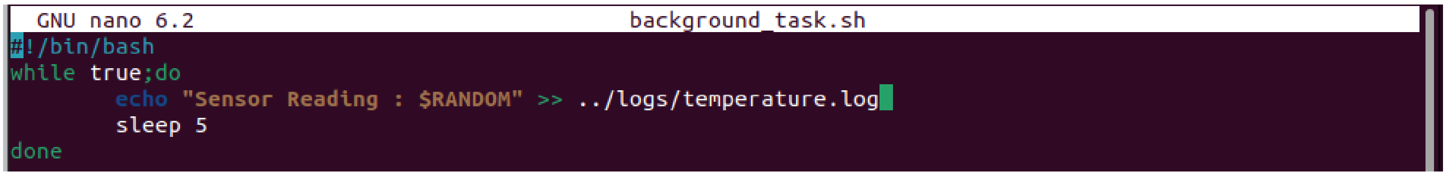
DAY4

Run a background task to simulate sensor polling.

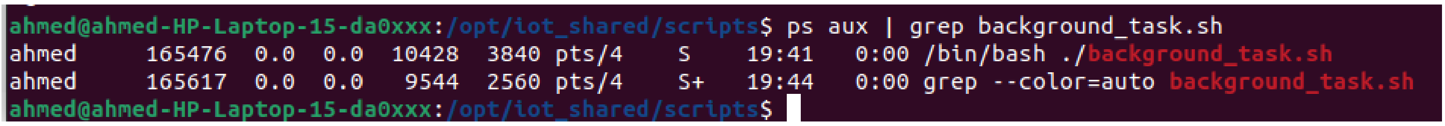


```
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared$ whoami
ahmed
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared$ ls
data  hard.log  logs  scripts  soft.log
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared$ cd scripts/
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ nano background_task.sh
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ ls ../logs
temperature.log
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ nano background_task.sh
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ chmod +x background_task.sh
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ ./background_task.sh &
[1] 165476
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ ps aux | grep background_task.sh
ahmed     165476  0.0  0.0  10428  3840 pts/4    S    19:41   0:00 /bin/bash ./background_ta
sk.sh
ahmed     165494  0.0  0.0   9544  2560 pts/4    S+   19:41   0:00 grep --color=auto backgro
und_task.sh
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ tail -f ../logs/temperature.log
Sensor Reading : 6445
Sensor Reading : 6384
Sensor Reading : 22692
Sensor Reading : 6318
```

List processes and filter for the background task.



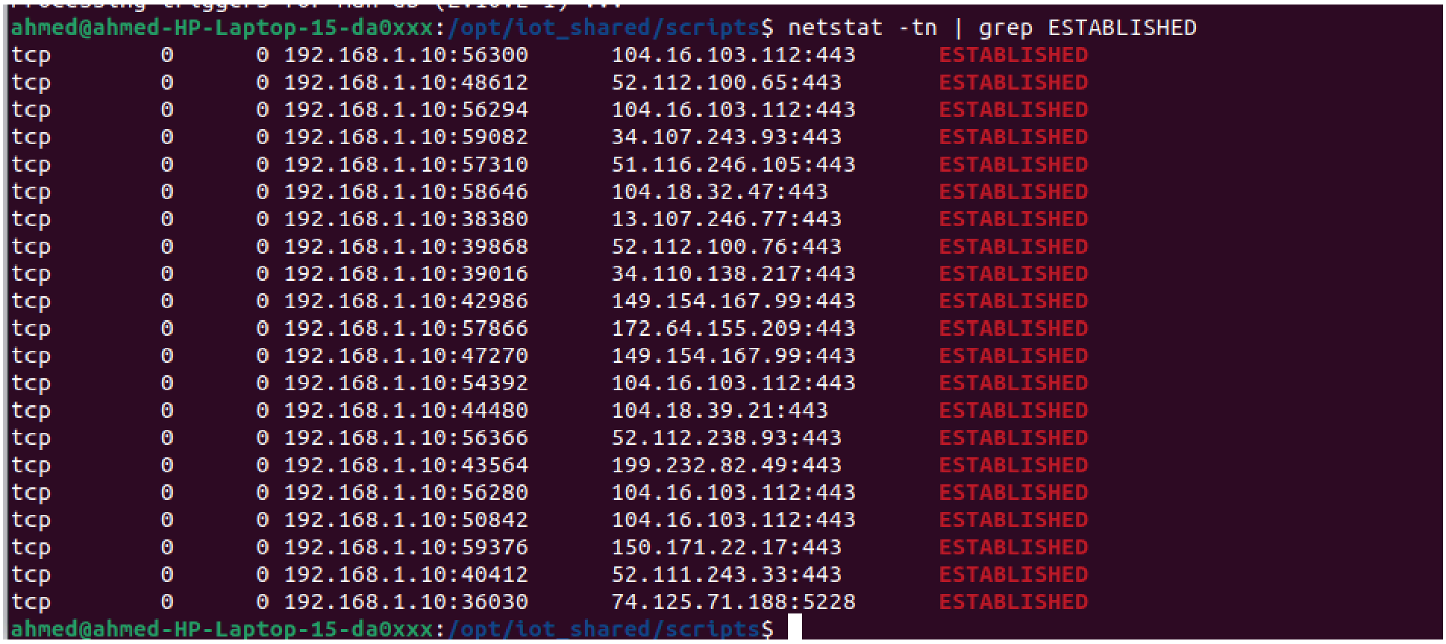
```
  GNU nano 6.2                         background_task.sh
#!/bin/bash
while true;do
        echo "Sensor Reading : $RANDOM" >> ../logs/temperature.log
        sleep 5
done
```



```
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ ps aux | grep background_task.sh
ahmed     165476  0.0  0.0  10428  3840 pts/4    S    19:41   0:00 /bin/bash ./background_task.sh
ahmed     165617  0.0  0.0   9544  2560 pts/4    S+   19:44   0:00 grep --color=auto background_task.sh
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$
```

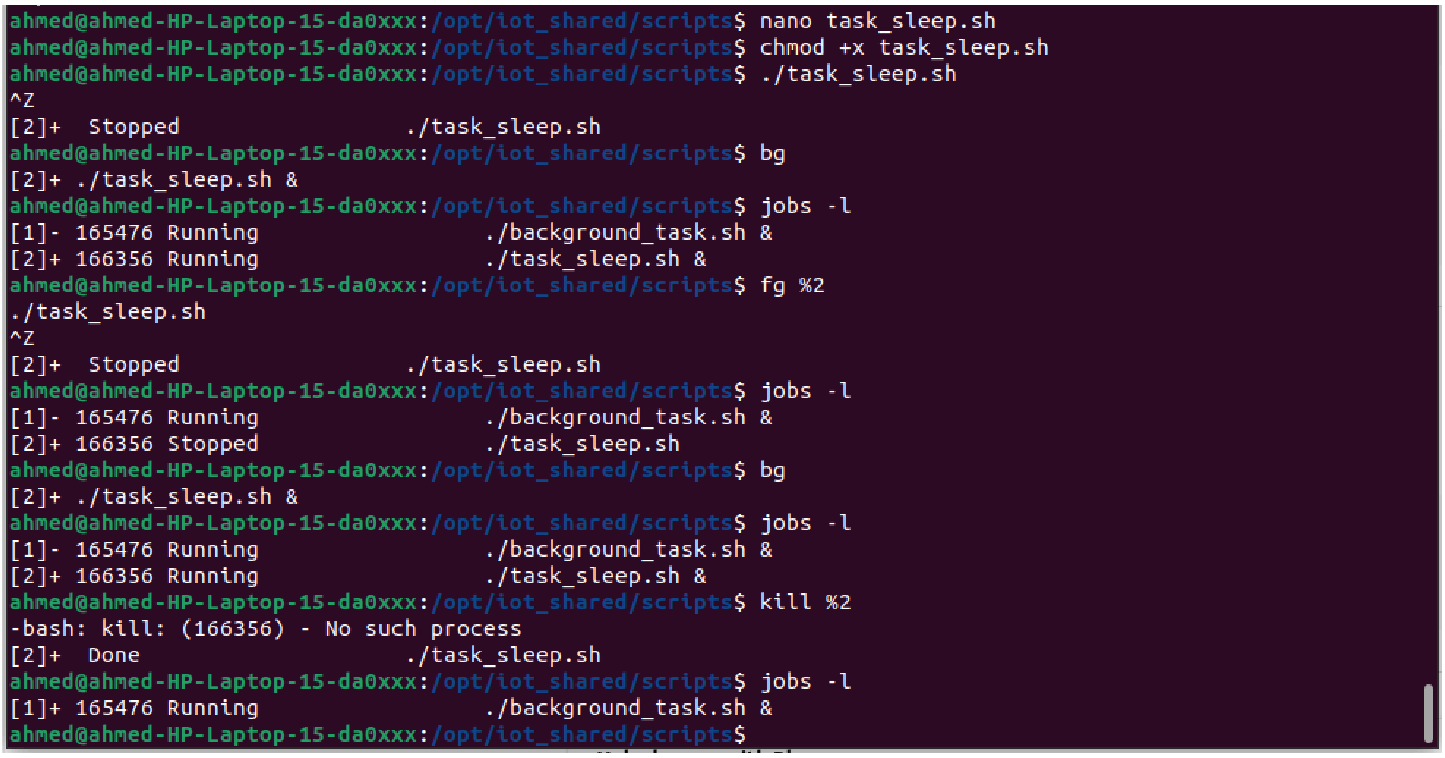
Check network states (established connections).

please don’t hack me 🙂

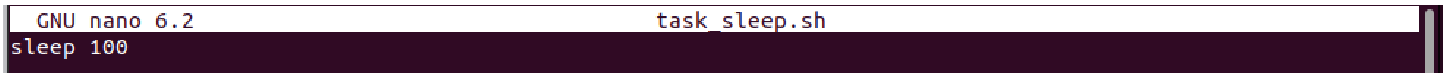


```
Processing triggers for man-db (2.10.2-1) ...
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ netstat -tn | grep ESTABLISHED
tcp        0      0 192.168.1.10:56300      104.16.103.112:443      ESTABLISHED
tcp        0      0 192.168.1.10:48612      52.112.100.65:443       ESTABLISHED
tcp        0      0 192.168.1.10:56294      104.16.103.112:443      ESTABLISHED
tcp        0      0 192.168.1.10:59082      34.107.243.93:443       ESTABLISHED
tcp        0      0 192.168.1.10:57310      51.116.246.105:443      ESTABLISHED
tcp        0      0 192.168.1.10:58646      104.18.32.47:443        ESTABLISHED
tcp        0      0 192.168.1.10:38380      13.107.246.77:443       ESTABLISHED
tcp        0      0 192.168.1.10:39868      52.112.100.76:443       ESTABLISHED
tcp        0      0 192.168.1.10:39016      34.110.138.217:443      ESTABLISHED
tcp        0      0 192.168.1.10:42986      149.154.167.99:443      ESTABLISHED
tcp        0      0 192.168.1.10:57866      172.64.155.209:443      ESTABLISHED
tcp        0      0 192.168.1.10:47270      149.154.167.99:443      ESTABLISHED
tcp        0      0 192.168.1.10:54392      104.16.103.112:443      ESTABLISHED
tcp        0      0 192.168.1.10:44480      104.18.39.21:443        ESTABLISHED
tcp        0      0 192.168.1.10:56366      52.112.238.93:443       ESTABLISHED
tcp        0      0 192.168.1.10:43564      199.232.82.49:443       ESTABLISHED
tcp        0      0 192.168.1.10:56280      104.16.103.112:443      ESTABLISHED
tcp        0      0 192.168.1.10:50842      104.16.103.112:443      ESTABLISHED
tcp        0      0 192.168.1.10:59376      150.171.22.17:443       ESTABLISHED
tcp        0      0 192.168.1.10:40412      52.111.243.33:443       ESTABLISHED
tcp        0      0 192.168.1.10:36030      74.125.71.188:5228      ESTABLISHED
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ 
```

Try foreground and background switching.

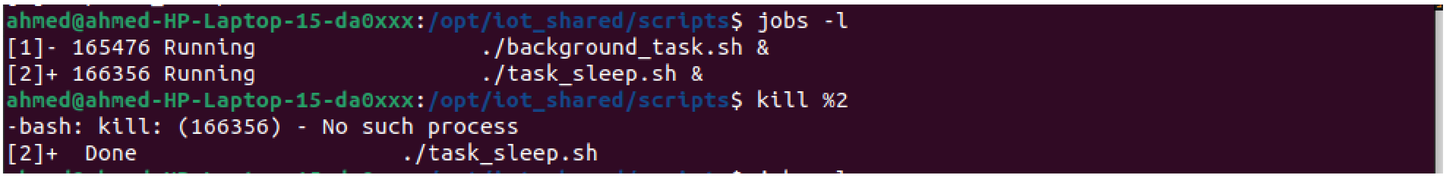


```
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ nano task_sleep.sh
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ chmod +x task_sleep.sh
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ ./task_sleep.sh
^Z
[2]+  Stopped                 ./task_sleep.sh
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ bg
[2]+ ./task_sleep.sh &
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ jobs -l
[1]- 165476 Running                 ./background_task.sh &
[2]+ 166356 Running                 ./task_sleep.sh &
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ fg %2
./task_sleep.sh
^Z
[2]+  Stopped                 ./task_sleep.sh
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ jobs -l
[1]- 165476 Running                 ./background_task.sh &
[2]+ 166356 Stopped                 ./task_sleep.sh
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ bg
[2]+ ./task_sleep.sh &
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ jobs -l
[1]- 165476 Running                 ./background_task.sh &
[2]+ 166356 Running                 ./task_sleep.sh &
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ kill %2
-bash: kill: (166356) - No such process
[2]+  Done                    ./task_sleep.sh
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ jobs -l
[1]+ 165476 Running                 ./background_task.sh &
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$
```



```
  GNU nano 6.2                                        task_sleep.sh
sleep 100
```

Kill a process if needed.



```
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ jobs -l
[1]- 165476 Running                 ./background_task.sh &
[2]+ 166356 Running                 ./task_sleep.sh &
ahmed@ahmed-HP-Laptop-15-da0xxx:/opt/iot_shared/scripts$ kill %2
-bash: kill: (166356) - No such process
[2]+  Done                    ./task_sleep.sh
```

Open-Ended Questions:

What happens step by step when you type a command in bash (e.g., ls) until you see the

output?

1- Parsing → parse the typed command and its flag from the bash

2- Mapping → map the typed command to its utility path and start to load the program

3- create process → parent process fork a new process (child process) in which we execute the utility of the ls

4- Terminate the process → child process prints the results and return 0 (success status) if process terminated successfully

---

Explain the types of processes in Linux: daemon, zombie, orphan. How can you detect

them?

Daemon → a background services that runs without user interaction.

How to detect ?

```
ps -ef | grep process-name
```

```
systemctl list-units --type=service
```

Zombie → A dead process whose execution is finished but still has an entry in the process table because its parent hasn't collected its exit status.

How to detect ?
ps aux | grep Z

Orphan → A process whose parent has exited but the process is still running.

How to detect ?

ps -ef | grep process-name

Why do we need Inter-Process Communication (IPC)? List some IPC mechanisms and real-life examples.

- each process has its own memory space and our process may interact with each other so we will need this communication to exchange data.
- Mechanisms → Sockets , Pipes , Message Queues and Shared Memory
- **Real-life use cases:**
  - Pipes: ls | grep
  - Shared memory: Database caching