

Day 4 – Phase 4: Process and Network Monitoring

Boss's Request: Secure the project and restrict access to authorized users only.

Tasks:

- Run a background task to simulate sensor polling.

```
sleep 100 &          # we can run any other process
```

- List processes and filter for the background task.

```
ps -f                # ps, ps -a, ps -e, ps aux  
# filter  
ps -f | grep sleep
```

```
mohamed@iot ~> sleep 100 &  
mohamed@iot ~> ps  
    PID TTY          TIME CMD  
  3503 pts/1        00:00:00 bash  
  3992 pts/1        00:00:01 fish  
  4819 pts/1        00:00:00 sleep  
  4829 pts/1        00:00:00 ps  
mohamed@iot ~> ps | grep sleep  
  4819 pts/1        00:00:00 sleep
```

- Check network states (established connections).

```
netstat | grep ESTABLISHED
```

```
mohamed@iot ~> netstat | grep ESTABLISHED
tcp        0      0 iot:ssh          192.168.100.35:52689 ESTABLISHED
tcp        0      0 iot:ssh          192.168.100.35:52688 ESTABLISHED
udp        0      0 iot:bootpc       _gateway:bootps    ESTABLISHED
```

- Try foreground and background switching.

- Run process
- `ctrl + z` → suspend
- `bg` → background
- `fg` → foreground

```
mohamed@iot ~> sleep 100
fish: Job 1, 'sleep 100' has stopped
mohamed@iot ~> bg
Send job 1 "sleep 100" to background
mohamed@iot ~> fg
Send job 1, "sleep 100" to foreground
```

- Kill a process if needed.

```
kill -9 <PID>
```

```
mohamed@iot ~> ps | grep sleep
 4887 pts/1    00:00:00 sleep
mohamed@iot ~> kill -9 4887
fish: Job 1, 'sleep 100' terminated by signal SIGKILL (Forced quit)
mohamed@iot ~> ps | grep sleep
```

Open-Ended Questions:

- What happens step by step when you type a command in bash (e.g., `ls`) until you see the output?
 - The **shell** interprets command.
 - Shell looks in `$PATH` to find `/bin/ls`.
 - **Kernel** loads program into memory as a process.
 - Process executes system calls to access the filesystem.
 - Output goes to `stdout` (terminal).
- Explain the types of processes in Linux: daemon, zombie, orphan. How can you detect them?
 - Daemon process
 - A long-running background process to respond to requests from services.
 - Orphan process
 - When a parent process ends first while the process is still running.
 - Zombie process
 - When a child has terminated but remains in the process table list.
- Why do we need Inter-Process Communication (IPC)? List some IPC mechanisms and real-life examples.
 - Processes have **parent-child** relationships and often need to exchange data.
 - IPC ensures communication.
 - Examples:
 - **Pipes (|)** → connect output of one process to input of another.
 - **Shared memory** → very fast data sharing.