Day 4 – Phase 4: Process and Network Monitoring

Boss’s Request: Monitor the system while simulating sensor activity.

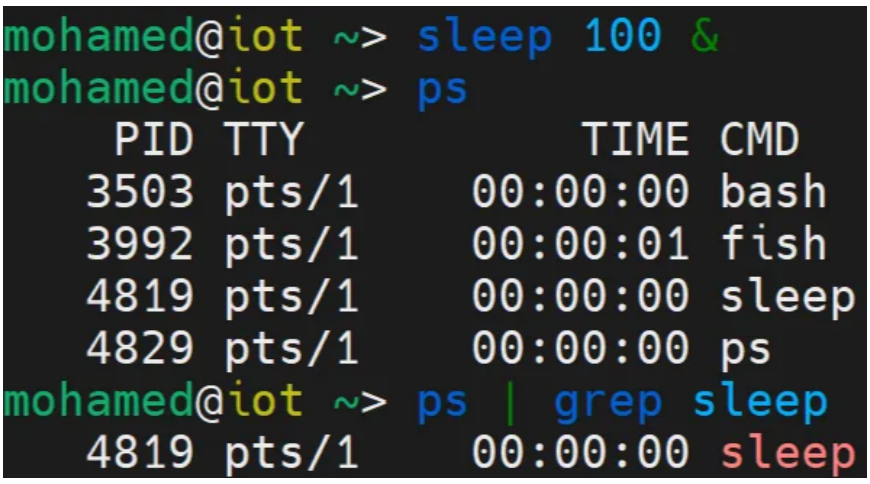
Tasks:

- Run a background task to simulate sensor polling.

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

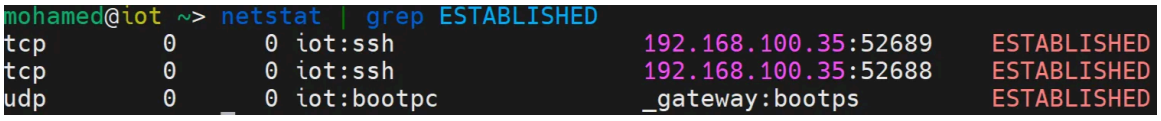
- List processes and filter for the background task.

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

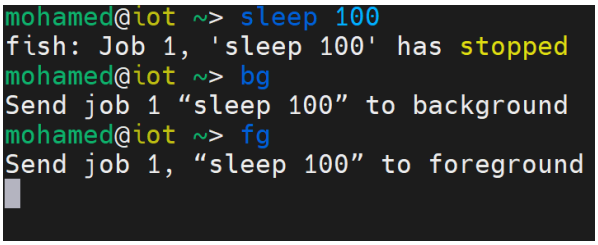


- Check network states (established connections).

```
netstat | grep ESTABLISHED
```

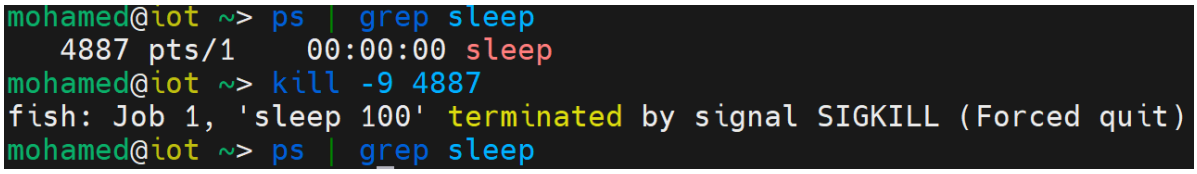


- Try foreground and background switching.
  - Run process
  - `ctrl + z` → suspend
  - `bg` → background
  - `fg` → foreground



- Kill a process if needed.

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



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.