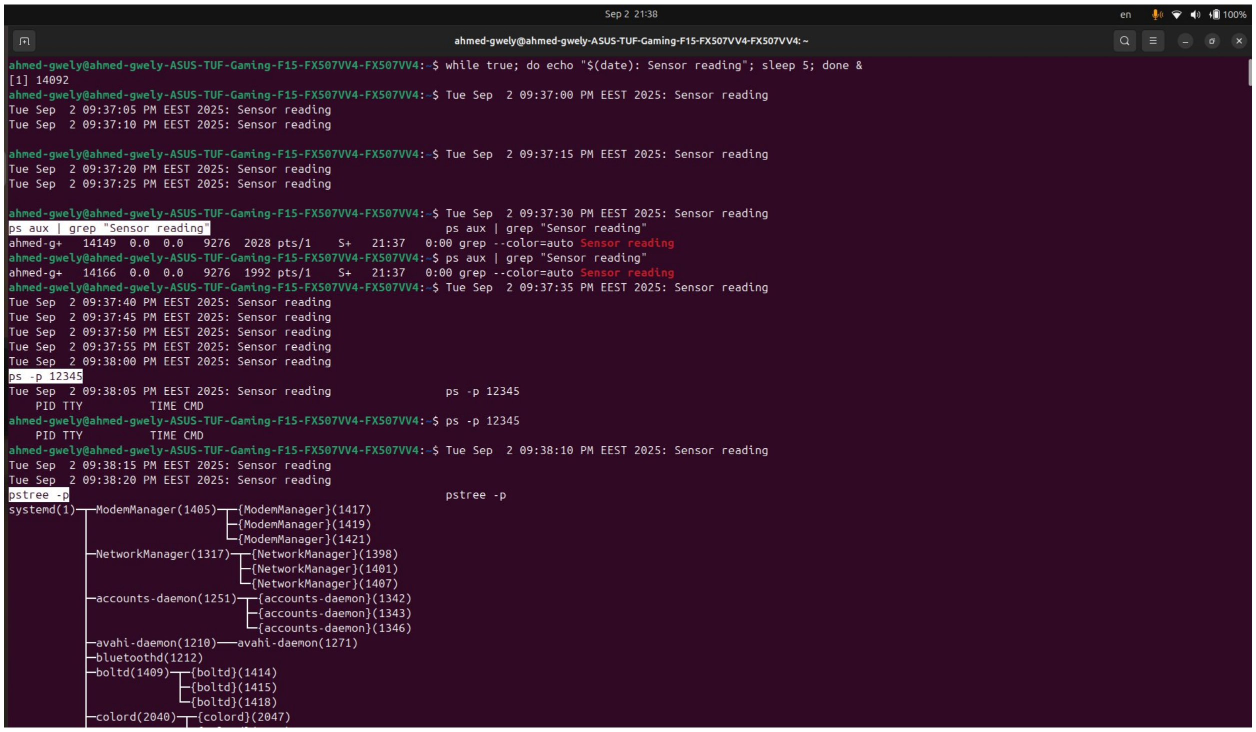
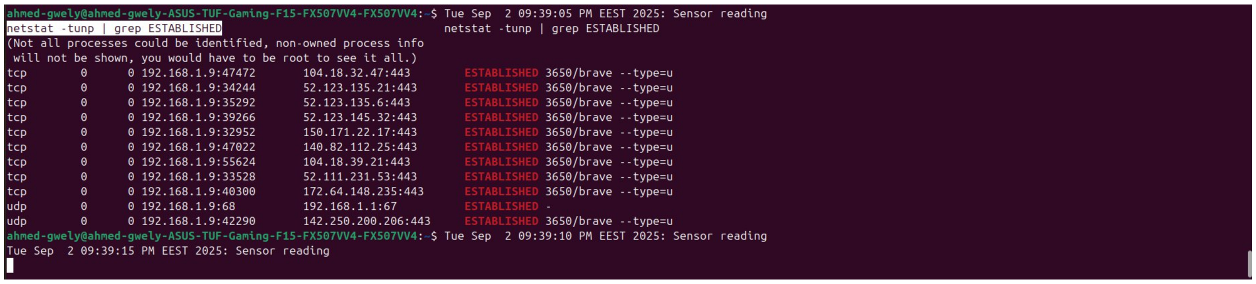
Day 4


Sep 2 21:38
en
100%
ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4: ~

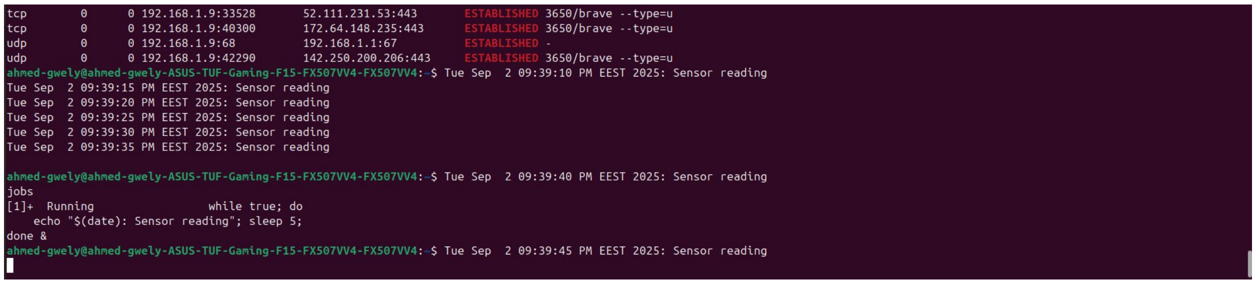
```
ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ while true; do echo "$(date): Sensor reading"; sleep 5; done &
[1] 14092
ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ Tue Sep  2 09:37:00 PM EEST 2025: Sensor reading
Tue Sep  2 09:37:05 PM EEST 2025: Sensor reading
Tue Sep  2 09:37:10 PM EEST 2025: Sensor reading

ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ Tue Sep  2 09:37:15 PM EEST 2025: Sensor reading
Tue Sep  2 09:37:20 PM EEST 2025: Sensor reading
Tue Sep  2 09:37:25 PM EEST 2025: Sensor reading

ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ Tue Sep  2 09:37:30 PM EEST 2025: Sensor reading
ps aux | grep "Sensor reading"                                   ps aux | grep "Sensor reading"
ahmed-g+   14149  0.0  0.0   9276  2028 pts/1    S+   21:37   0:00 grep --color=auto Sensor reading
ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ ps aux | grep "Sensor reading"
ahmed-g+   14166  0.0  0.0   9276  1992 pts/1    S+   21:37   0:00 grep --color=auto Sensor reading
ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ Tue Sep  2 09:37:35 PM EEST 2025: Sensor reading
Tue Sep  2 09:37:40 PM EEST 2025: Sensor reading
Tue Sep  2 09:37:45 PM EEST 2025: Sensor reading
Tue Sep  2 09:37:50 PM EEST 2025: Sensor reading
Tue Sep  2 09:37:55 PM EEST 2025: Sensor reading
Tue Sep  2 09:38:00 PM EEST 2025: Sensor reading
ps -p 12345
Tue Sep  2 09:38:05 PM EEST 2025: Sensor reading                 ps -p 12345
    PID TTY          TIME CMD
ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ ps -p 12345
    PID TTY          TIME CMD
ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ Tue Sep  2 09:38:10 PM EEST 2025: Sensor reading
Tue Sep  2 09:38:15 PM EEST 2025: Sensor reading
Tue Sep  2 09:38:20 PM EEST 2025: Sensor reading
pstree -p                                                        pstree -p
systemd(1)─┬─ModemManager(1405)─┬─{ModemManager}(1417)
           │                    ├─{ModemManager}(1419)
           │                    └─{ModemManager}(1421)
           ├─NetworkManager(1317)─┬─{NetworkManager}(1398)
           │                      ├─{NetworkManager}(1401)
           │                      └─{NetworkManager}(1407)
           ├─accounts-daemon(1251)─┬─{accounts-daemon}(1342)
           │                       ├─{accounts-daemon}(1343)
           │                       └─{accounts-daemon}(1346)
           ├─avahi-daemon(1210)───avahi-daemon(1271)
           ├─bluetoothd(1212)
           ├─boltd(1409)─┬─{boltd}(1414)
           │             ├─{boltd}(1415)
           │             └─{boltd}(1418)
           ├─colord(2040)─┬─{colord}(2047)
```

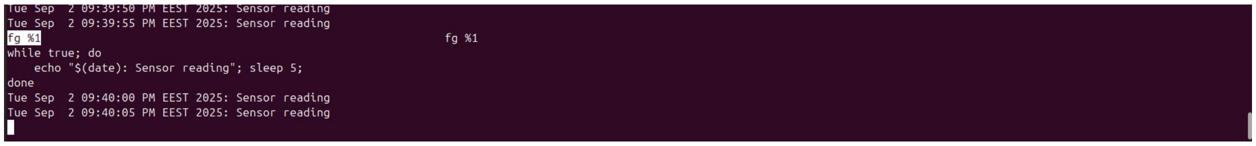


```
ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ Tue Sep  2 09:39:05 PM EEST 2025: Sensor reading
netstat -tunp | grep ESTABLISHED                                 netstat -tunp | grep ESTABLISHED
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 192.168.1.9:47472       104.18.32.47:443        ESTABLISHED 3650/brave --type=u
tcp        0      0 192.168.1.9:34244       52.123.135.21:443       ESTABLISHED 3650/brave --type=u
tcp        0      0 192.168.1.9:35292       52.123.135.6:443        ESTABLISHED 3650/brave --type=u
tcp        0      0 192.168.1.9:39266       52.123.145.32:443       ESTABLISHED 3650/brave --type=u
tcp        0      0 192.168.1.9:32952       150.171.22.17:443       ESTABLISHED 3650/brave --type=u
tcp        0      0 192.168.1.9:47022       140.82.112.25:443       ESTABLISHED 3650/brave --type=u
tcp        0      0 192.168.1.9:55624       104.18.39.21:443        ESTABLISHED 3650/brave --type=u
tcp        0      0 192.168.1.9:33528       52.111.231.53:443       ESTABLISHED 3650/brave --type=u
tcp        0      0 192.168.1.9:40300       172.64.148.235:443      ESTABLISHED 3650/brave --type=u
udp        0      0 192.168.1.9:68          192.168.1.1:67          ESTABLISHED -
udp        0      0 192.168.1.9:42290       142.250.200.206:443     ESTABLISHED 3650/brave --type=u
ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ Tue Sep  2 09:39:10 PM EEST 2025: Sensor reading
Tue Sep  2 09:39:15 PM EEST 2025: Sensor reading
█
```

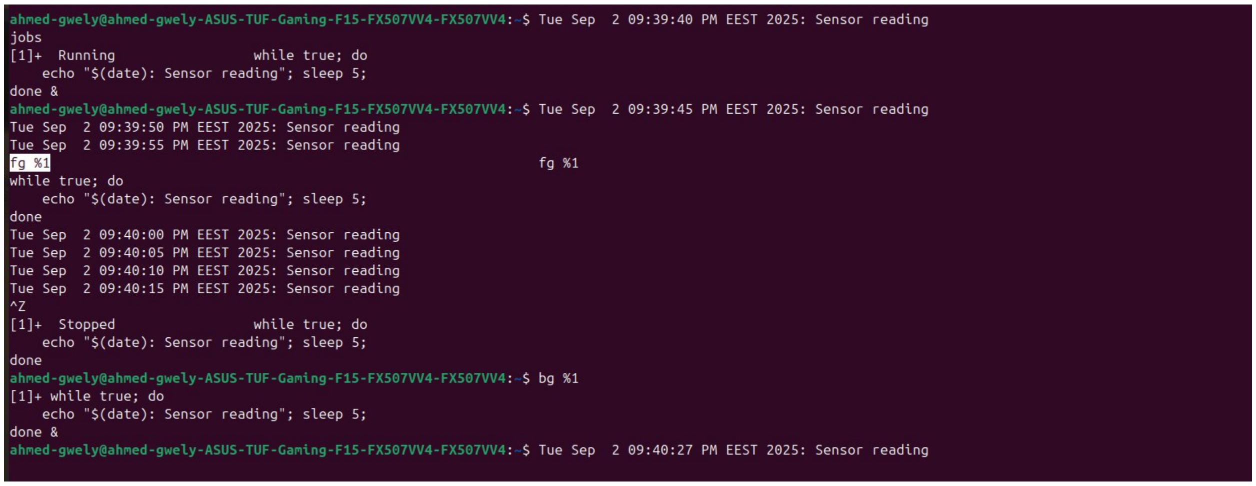


```
tcp        0      0 192.168.1.9:33528       52.111.231.53:443       ESTABLISHED 3650/brave --type=u
tcp        0      0 192.168.1.9:40300       172.64.148.235:443      ESTABLISHED 3650/brave --type=u
udp        0      0 192.168.1.9:68          192.168.1.1:67          ESTABLISHED -
udp        0      0 192.168.1.9:42290       142.250.200.206:443     ESTABLISHED 3650/brave --type=u
ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ Tue Sep  2 09:39:10 PM EEST 2025: Sensor reading
Tue Sep  2 09:39:15 PM EEST 2025: Sensor reading
Tue Sep  2 09:39:20 PM EEST 2025: Sensor reading
Tue Sep  2 09:39:25 PM EEST 2025: Sensor reading
Tue Sep  2 09:39:30 PM EEST 2025: Sensor reading
Tue Sep  2 09:39:35 PM EEST 2025: Sensor reading

ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ Tue Sep  2 09:39:40 PM EEST 2025: Sensor reading
jobs
[1]+  Running                 while true; do
    echo "$(date): Sensor reading"; sleep 5;
done &
ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ Tue Sep  2 09:39:45 PM EEST 2025: Sensor reading
█
```



```
Tue Sep  2 09:39:50 PM EEST 2025: Sensor reading
Tue Sep  2 09:39:55 PM EEST 2025: Sensor reading
fg %1                                                    fg %1
while true; do
    echo "$(date): Sensor reading"; sleep 5;
done
Tue Sep  2 09:40:00 PM EEST 2025: Sensor reading
Tue Sep  2 09:40:05 PM EEST 2025: Sensor reading
█
```



```
ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ Tue Sep  2 09:39:40 PM EEST 2025: Sensor reading
jobs
[1]+  Running                 while true; do
    echo "$(date): Sensor reading"; sleep 5;
done &
ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ Tue Sep  2 09:39:45 PM EEST 2025: Sensor reading
Tue Sep  2 09:39:50 PM EEST 2025: Sensor reading
Tue Sep  2 09:39:55 PM EEST 2025: Sensor reading
fg %1                                                            fg %1
while true; do
    echo "$(date): Sensor reading"; sleep 5;
done
Tue Sep  2 09:40:00 PM EEST 2025: Sensor reading
Tue Sep  2 09:40:05 PM EEST 2025: Sensor reading
Tue Sep  2 09:40:10 PM EEST 2025: Sensor reading
Tue Sep  2 09:40:15 PM EEST 2025: Sensor reading
^Z
[1]+  Stopped                 while true; do
    echo "$(date): Sensor reading"; sleep 5;
done
ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ bg %1
[1]+ while true; do
    echo "$(date): Sensor reading"; sleep 5;
done &
ahmed-gwely@ahmed-gwely-ASUS-TUF-Gaming-F15-FX507VV4-FX507VV4:~$ Tue Sep  2 09:40:27 PM EEST 2025: Sensor reading
```

What happens step by step when you type a command in Bash (`ls`)

1. **Input:**
    - You type `ls` and press Enter in the Bash shell.
1. **Parsing:**
    - Bash parses your input to check for commands, arguments, pipes, redirections, or variables.
1. **Command search:**
    - Bash looks for the command in:
        1. **Aliases/functions** in the shell
        2. **Built-in commands** (like `cd`)
        3. **Executable files in `$PATH`** directories (`/bin`, `/usr/bin`, etc.)
2. **Fork and exec:**
    - Bash **forks** a new process (child).
    - The child process **executes** the command (replaces its memory with the command's executable).
1. **Process scheduling:**
    - Linux kernel schedules the new process to run on the CPU.

1. **Execution:**
    - The `ls` process reads the directory contents, formats output, etc.
1. **Output:**
    - The process writes the result to **stdout**, which the terminal displays.
1. **Termination:**
    - The process exits, returning an **exit code** to the parent (Bash).
    - Bash shows a prompt again for the next command.

Types of processes in Linux

Process Type	**Description**	**How to detect**
Daemon	Background service, usually starts at boot and runs without a terminal (e.g., `sshd`, `cron`)	`ps aux
Zombie	Process that has finished execution but still has an entry in the **process table** because the parent hasn't read its exit status	`ps aux
Orphan	Process whose parent has terminated; adopted by `init` (PID 1)	`ps -ef

Why do we need Inter-Process Communication (IPC)?

- **Reason:** Processes are isolated in Linux; to work together, they need a way to **exchange data or signals**.
- Without IPC, each process would be completely independent.

IPC mechanisms:

IPC Method	**Description**	**Real-life example**
**Pipe (`	`)**	Unidirectional data stream between processes
Named Pipe (FIFO)	Pipe with a name, can be used between unrelated processes	Logging system writing to a common FIFO file
Signals	Send a notification to a process (e.g., terminate, stop)	`kill -SIGTERM <PID>`
Shared Memory	Multiple processes access the same memory segment	High-speed trading apps sharing live market data
Message Queue	Queue of messages for processes	Print spooler queue (`lp`/`cups`)
Sockets	Communication over network (or locally)	Client-server apps, web browsers