Assignment 2 — Introduction & Purpose

1. What is this assignment about?

This lab will give you practical, hands-on exposure to how Linux reports information about your system's hardware, resources, and processes.
You will:

- Explore the **`/proc` filesystem**, which is a **real-time view** into the kernel's data structures — no physical files, just live system information.
- Use **core Linux utilities** (`ps`, `lscpu`, `df`, `ip`, `lsblk`, `strace`, etc.) to gather system statistics.
- Practice **controlling processes** with UNIX signals (`SIGSTOP`, `SIGCONT`).
- Learn to **trace system calls** made by programs using `strace`.

By the end, you'll be able to:

- Identify your system's CPU, memory, disk, and network configuration.
- Inspect how many processes are running and what they are doing.
- Monitor I/O usage and network statistics.
- Understand which **system calls** a program uses to interact with the OS.

2. Why is this important in Operating Systems?

This is where OS theory meets reality.
Here's what you're actually learning:

1. **Kernel ↔ User Space Bridge**
   - `/proc` is your window into kernel space. You're reading the *same* data the OS scheduler, memory manager, and device drivers use.
2. **Direct System Monitoring**
   - No pretty GUI — you will see the raw numbers the kernel tracks for CPU load, context switches, memory usage, and device activity.
3. **Live Application of OS Concepts**
   - Terms like *context switch*, *load average*, *system call*, and *process state* become concrete when you measure them yourself.
4. **Real Process Control**
   - Sending signals to stop and resume a process lets you simulate what the OS does internally.

5. **System Call Awareness**

    - `strace` will reveal exactly *how* a user program asks the OS to open files, read/write data, or use the network.

3. How is this useful in real life?

You’ll use these skills in many scenarios:

Field / Role	How this lab helps
System Administration / DevOps	Diagnose high CPU/memory usage, spot I/O bottlenecks, track misbehaving processes.
Performance Engineering	Measure resource usage, identify slow or resource-hungry code.
Debugging & Troubleshooting	See why a process is stuck by checking open files, sockets, and syscalls.
Security & Forensics	Detect suspicious processes or unexpected network activity.
Systems Programming	Know which syscalls your program will make and how they behave.
Job Interviews	`/proc`, process states, system calls, and Linux monitoring commands are common OS interview topics.

4. Core Concepts You Should Know Before Starting

1. **The `/proc` Filesystem**

    - A *virtual filesystem* — nothing stored on disk.
    - Files like `/proc/cpuinfo`, `/proc/meminfo`, `/proc/<PID>/status` are *generated on the fly* when you read them.
    - Used by commands like `top`, `ps`, and `free` behind the scenes.

2. **Process IDs (PIDs)**

    - Each process has a unique ID.
    - Used to inspect (`ps`, `cat /proc/<PID>/status`) or control (`kill`) processes.

3. **System Calls**

    - Low-level OS functions like `open()`, `read()`, `write()` that programs use to access hardware, files, and the network.
    - User programs can’t talk to hardware directly — they must use syscalls via the kernel.

4. **Signals**

- Asynchronous notifications to a process.
- Examples:
  `SIGSTOP` → pause execution
  `SIGCONT` → resume execution
  `SIGKILL` → terminate immediately

5. **Basic Linux Commands**
   - Viewing files: `cat`, `less`, `head`
   - Searching text: `grep`
   - Process listing: `ps`, `top`
   - Memory/disk info: `free`, `df`
   - Network info: `ip`, `ss`

Short Cheat-sheet

- CPU summary: `lscpu`
- Logical CPUs: `grep -c '^processor' /proc/cpuinfo`
- Physical sockets: `awk '/physical id/ {ids[$NF]=1} END{print length(ids)}' /proc/cpuinfo`
- Model name: `awk -F: '/model name/ {print $2; exit}' /proc/cpuinfo`
- CPU frequency: `lscpu | grep MHz` or `grep 'cpu MHz' /proc/cpuinfo`
- Memory: `free -h` / `grep -E 'MemTotal|MemFree|MemAvailable' /proc/meminfo`
- Swap: `swapon --show` / `cat /proc/swaps`
- Kernel: `uname -r`
- Processes: `ps -e --no-headers | wc -l`
- Context switches: `awk '/^ctxt/ {print $2}' /proc/stat`
- Uptime: `uptime` / `cat /proc/uptime`
- Disk I/O: `cat /proc/diskstats` ; `iostat -dx` (optional)
- Filesystem usage: `df -hT`

- Block devices: `lsblk -o NAME,SIZE,FSTYPE,MOUNTPOINT`
- Net stats: `cat /proc/net/dev` ; `ip addr show`
- FDs: `ls -l /proc/<PID>/fd` ; `lsof -p <PID>`
- Stop/Continue: `kill -STOP <PID>` ; `kill -CONT <PID>`
- strace: `strace -o out.txt -f <command>`
- Redirect capture: `cmd > outfile 2>&1`