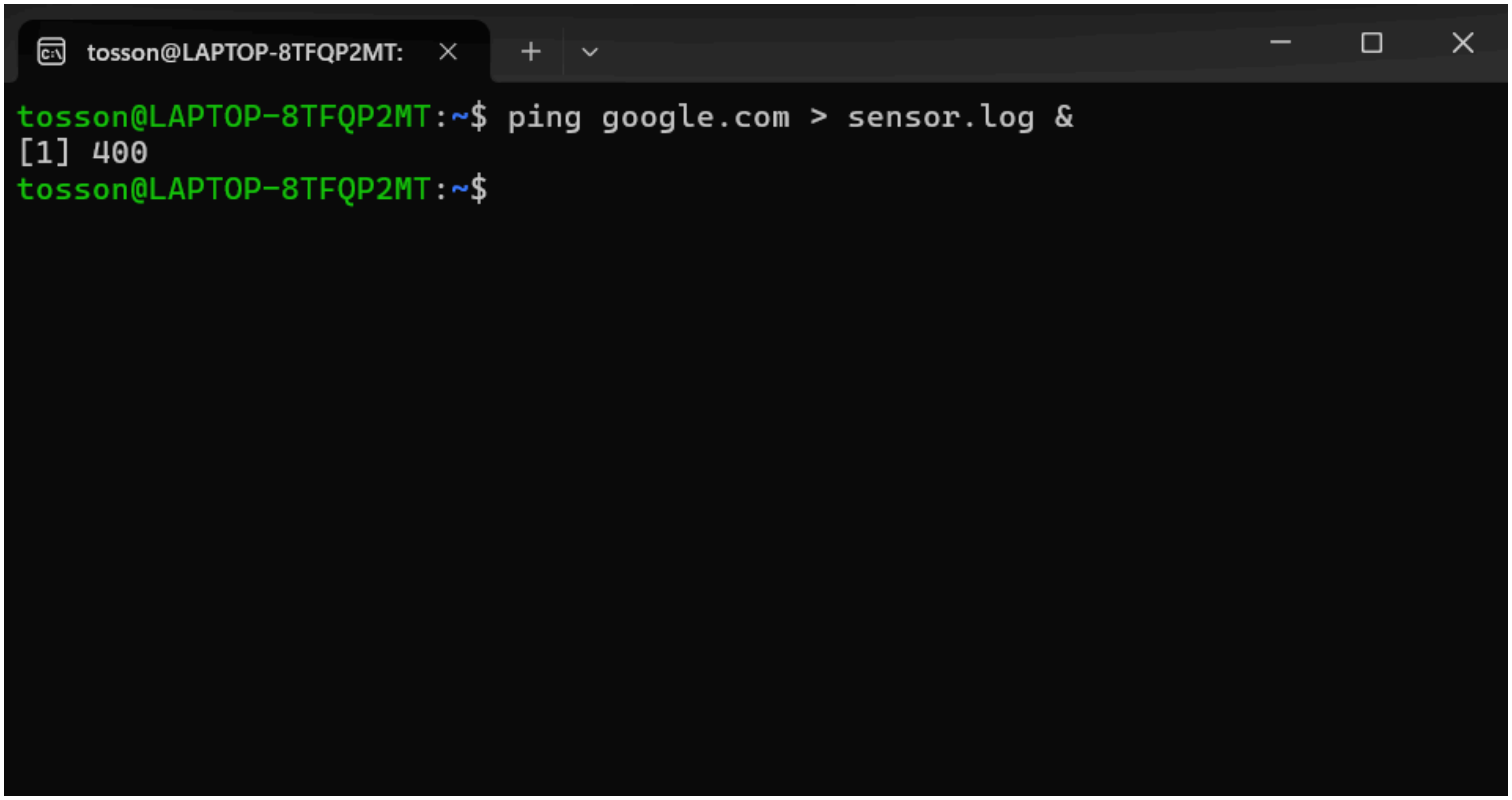
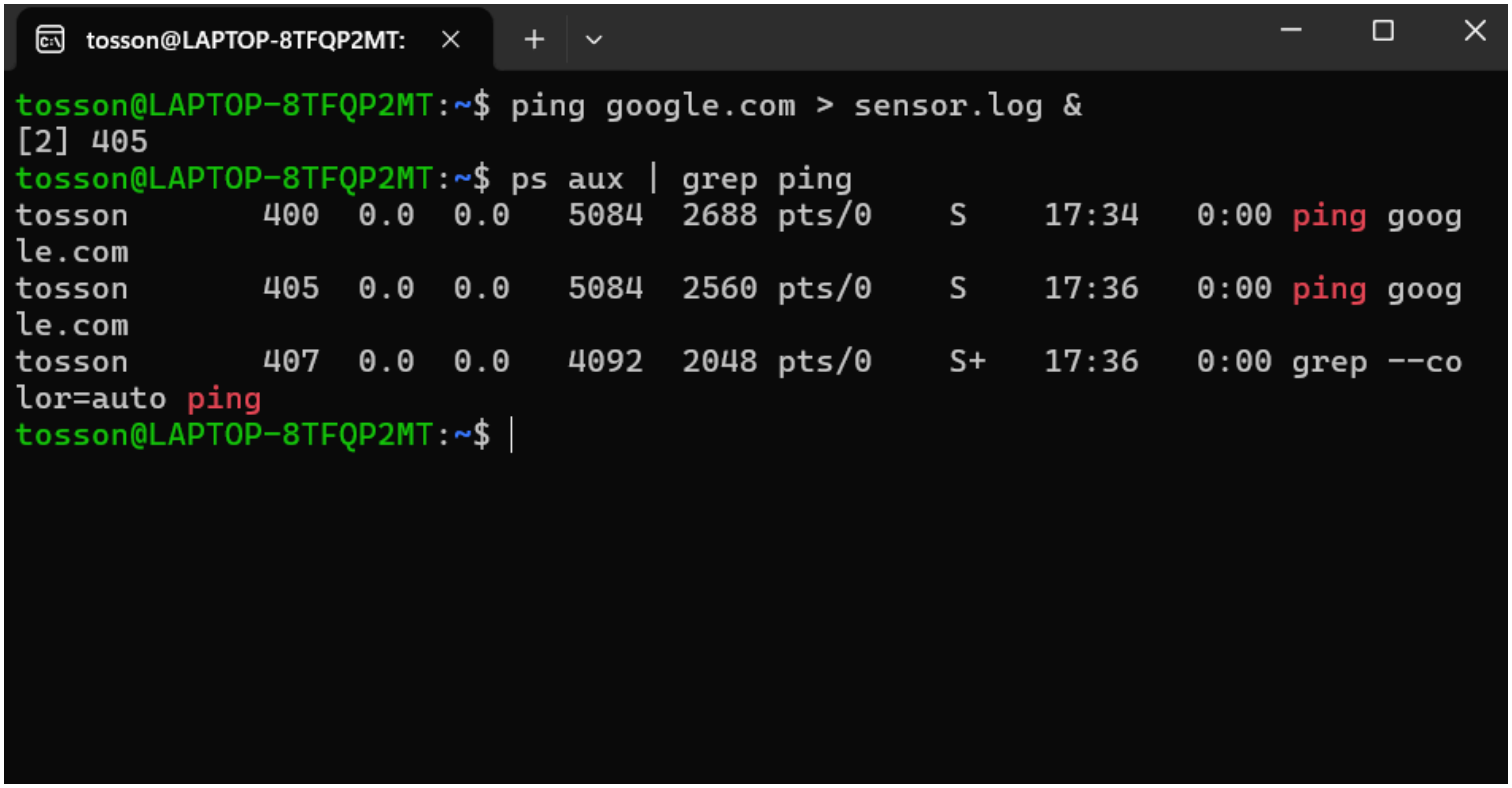
Day4

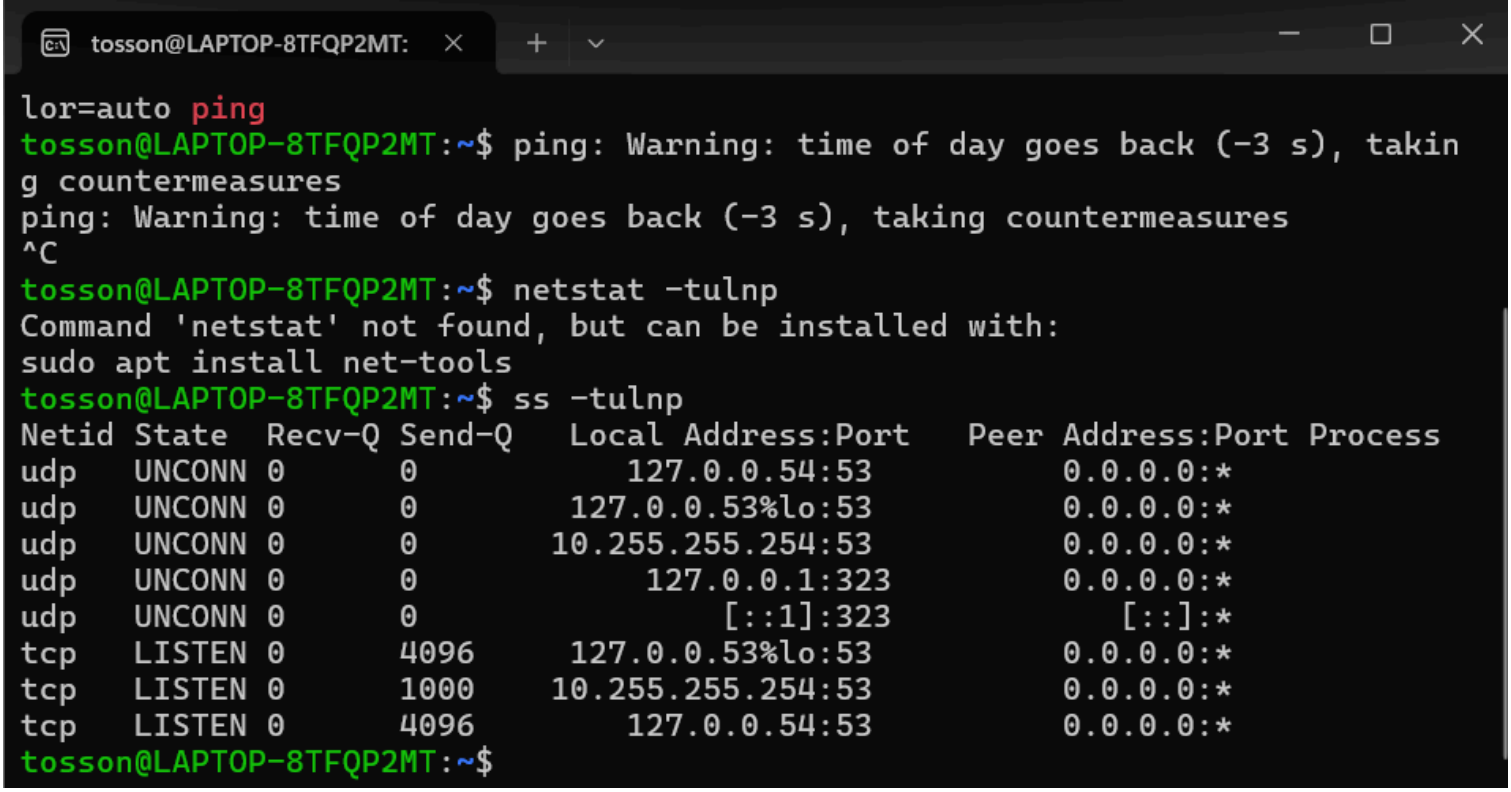
Run a background task to simulate sensor polling.



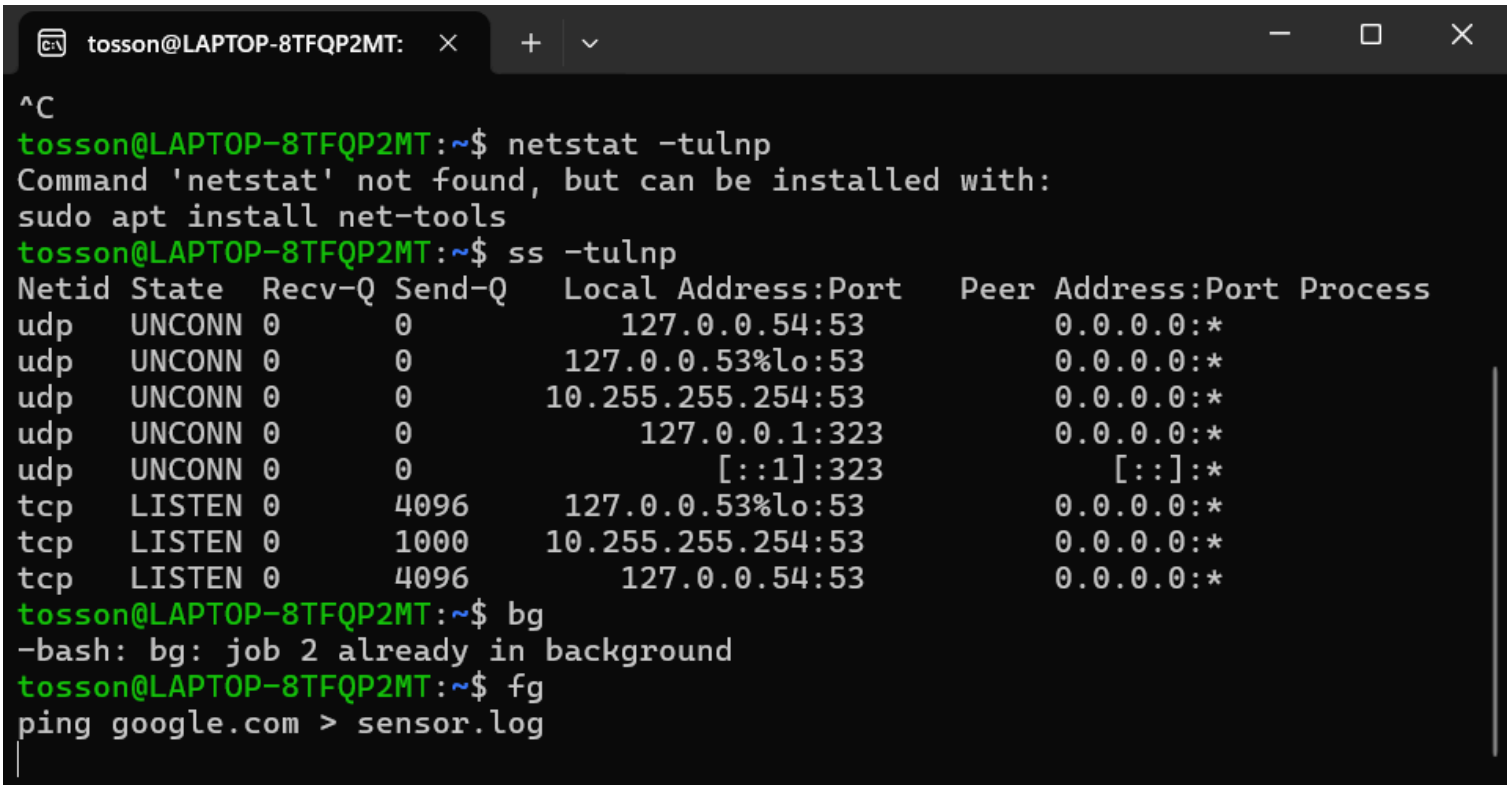
List processes and filter for the background task.



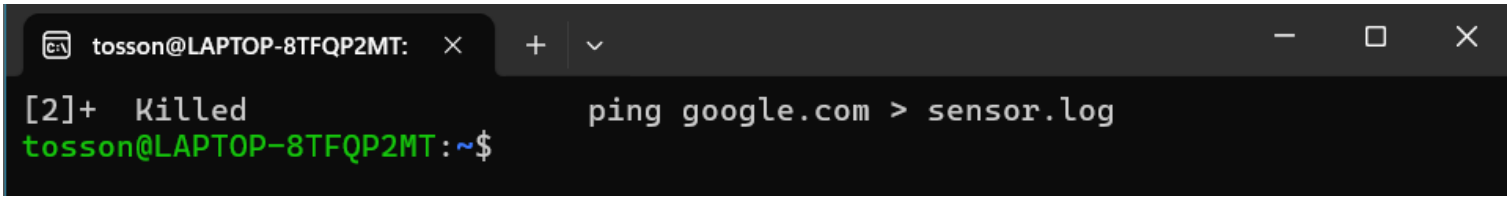
Check network states (established connections).



Try foreground and background switching.



Kill a process if needed



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

1. **Shell reads** the line you typed.
2. **Parses & expands** variables, globs, command substitutions.
3. **Resolves** the name (builtin/function/alias or external via `$PATH`).
4. **Forks** a child process (for external commands).
5. Child **applies redirections/pipes** then `execve()`s the program.
6. Program **runs** and writes to `stdout` (the terminal).
7. Program **exits**; the shell collects its status and shows the prompt.

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

daemon is a background process (like `sshd`) that runs without user interaction; you can spot it with `ps -ef` and usually its terminal (TTY) is `?`. A zombie process is one that finished running but its parent hasn't cleaned it up yet; it shows as `Z` in `ps` under the stat column. An orphan process is one whose parent died, so it gets adopted by `init` (PID 1); you can see this in `ps -ef` if the parent PID is `1`.

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

IPC is needed so processes can share data and coordinate. Methods include **pipes** (`ls | grep`), message queues (chat apps), shared memory (databases), and sockets (browsers talking to servers). This lets programs work together smoothly.