

# Day4

Run a background task to simulate sensor polling.

```
tosson@LAPTOP-8TFQP2MT: ~$ ping google.com > sensor.log &
[1] 400
tosson@LAPTOP-8TFQP2MT: ~$
```

List processes and filter for the background task.

```
tosson@LAPTOP-8TFQP2MT: ~$ ping google.com > sensor.log &
[2] 405
tosson@LAPTOP-8TFQP2MT: ~$ ps aux | grep ping
tosson      400  0.0  0.0   5084  2688 pts/0    S   17:34   0:00 ping goog
le.com
tosson      405  0.0  0.0   5084  2560 pts/0    S   17:36   0:00 ping goog
le.com
tosson      407  0.0  0.0   4092  2048 pts/0    S+  17:36   0:00 grep --co
lor=auto ping
tosson@LAPTOP-8TFQP2MT: ~$ |
```

Check network states (established connections).

```
tosson@LAPTOP-8TFQP2MT: x + v
lor=auto ping
tosson@LAPTOP-8TFQP2MT:~$ ping: Warning: time of day goes back (-3 s), taking countermeasures
ping: Warning: time of day goes back (-3 s), taking countermeasures
^C
tosson@LAPTOP-8TFQP2MT:~$ netstat -tulnp
Command 'netstat' not found, but can be installed with:
sudo apt install net-tools
tosson@LAPTOP-8TFQP2MT:~$ ss -tulnp
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port Process
udp UNCONN 0 0 127.0.0.54:53 0.0.0.0:*
udp UNCONN 0 0 127.0.0.53%lo:53 0.0.0.0:*
udp UNCONN 0 0 10.255.255.254:53 0.0.0.0:*
udp UNCONN 0 0 127.0.0.1:323 0.0.0.0:*
udp UNCONN 0 0 [::1]:323 [::]:*
tcp LISTEN 0 4096 127.0.0.53%lo:53 0.0.0.0:*
tcp LISTEN 0 1000 10.255.255.254:53 0.0.0.0:*
tcp LISTEN 0 4096 127.0.0.54:53 0.0.0.0:*
tosson@LAPTOP-8TFQP2MT:~$
```

Try foreground and background switching.

```
tosson@LAPTOP-8TFQP2MT: x + v
^C
tosson@LAPTOP-8TFQP2MT:~$ netstat -tulnp
Command 'netstat' not found, but can be installed with:
sudo apt install net-tools
tosson@LAPTOP-8TFQP2MT:~$ ss -tulnp
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port Process
udp UNCONN 0 0 127.0.0.54:53 0.0.0.0:*
udp UNCONN 0 0 127.0.0.53%lo:53 0.0.0.0:*
udp UNCONN 0 0 10.255.255.254:53 0.0.0.0:*
udp UNCONN 0 0 127.0.0.1:323 0.0.0.0:*
udp UNCONN 0 0 [::1]:323 [::]:*
tcp LISTEN 0 4096 127.0.0.53%lo:53 0.0.0.0:*
tcp LISTEN 0 1000 10.255.255.254:53 0.0.0.0:*
tcp LISTEN 0 4096 127.0.0.54:53 0.0.0.0:*
tosson@LAPTOP-8TFQP2MT:~$ bg
-bash: bg: job 2 already in background
tosson@LAPTOP-8TFQP2MT:~$ fg
ping google.com > sensor.log
|
```

Kill a process if needed

```
tosson@LAPTOP-8TFQP2MT: x + v
[2]+ Killed ping google.com > sensor.log
tosson@LAPTOP-8TFQP2MT:~$
```

# 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.