🧠 `ps` Command in Linux — Complete Cheat Sheet (with Examples, Use Cases & Developer Tips)

The `ps` command displays **a snapshot of current processes**. To monitor them live, use `top` or `htop`.

⚙️ Option Styles Accepted by `ps`

1. **UNIX options**: grouped with a single dash `-` Example: `ps -ef`
2. **BSD options**: grouped without a dash Example: `ps aux`
3. **GNU long options**: start with two dashes Example: `ps --sort=-%cpu`

> These **cannot be mixed together** in a single command.

📌 BASIC SYNTAX

```
ps [UNIX or BSD or GNU options]
```

📌 ESSENTIAL COMMANDS

Show all processes (standard style):

```
ps -e          # All processes
ps -ef         # Full-format (with UID, PID, PPID, CMD, etc.)
ps -eF         # Extra full format (includes NI, PRI, SZ)
ps -ely        # Long format with priority info
```

Show all processes (BSD style):

```
ps ax          # All processes
ps axu         # All with user ownership info
```

🌳 Process Tree (Parent → Child relationships)

```
ps -ejH        # System V style with hierarchical output
ps axjf        # BSD style job format tree
```

> 🔍 Useful to trace processes spawned by scripts or services

🧵 Thread Information

```
ps -eLf        # Threads using LWP (Lightweight Process)
ps axms        # BSD multi-threaded summary
```

> Good for debugging multi-threaded apps (Java, Python threading, etc.)

🔐 Security-Related Information

```
ps -eo euser,ruser,suser,fuser,f,comm,label
ps axZ          # Shows security context (SELinux)
ps -eM          # Security module info
```

🔍 Show All Root Processes (Real & Effective User IDs)

```
ps -U root -u root u
```

✍️ Custom Output Formatting

```
ps -eo pid,tid,class,rtprio,ni,pri,psr,pcpu,stat,wchan:14,comm
ps axo stat,euid,ruid,tty,tpgid,sess,pgrp,ppid,pid,pcpu,comm
ps -Ao pid,tt,user,fname,tmout,f,wchan
```

> Use `-o` or `-eo` for field selection. Ideal for scripting/logging.

🔍 Filter Specific Process

Show only PID of a process:

```
ps -C syslogd -o pid=
```

Get process name from PID:

```
ps -q 42 -o comm=
```

> These are great for filtering in scripts or system monitoring tools.

🧠 DEVELOPER-FRIENDLY USE CASES

🔹 List top memory-consuming processes:

```
ps aux --sort=-%mem | head -10
```

🔹 List top CPU-consuming processes:

```
ps -eo pid,comm,%cpu,%mem --sort=-%cpu | head -10
```

🔹 Monitor a specific process live:

```
watch -n 2 "ps -p <pid> -o pid,ppid,%cpu,%mem,cmd"
```

🔹 Get all daemons (background processes with no TTY):

```
ps -eo pid,tty,stat,cmd | grep '?'
```

Find zombie processes:

```
ps aux | awk '$8=="Z"'
```

Kill a process running on a specific port:

```
sudo lsof -i :<port>
ps -p <pid>
kill <pid>
```

PRO TIPS FOR SCRIPTS & MONITORING

Periodic Logging of All Running Processes:

```
while true; do ps -eo pid,cmd,%cpu,%mem >> /var/log/ps-log.txt; sleep 60; done
```

Show all Java-based background services:

```
ps aux | grep java
```

List threads of a PID (useful for debugging):

```
ps -L -p <pid>
```

PROCESS STATUSES (`STAT` column)

Code	Meaning
R	Running
S	Sleeping
D	Uninterruptible sleep (e.g., IO)
Z	Zombie
T	Traced or stopped
X	Dead
W	Paging
<	High priority
N	Low priority
+	Foreground group

Code	Meaning

SUMMARY TABLE

Task	Command
All processes (SysV)	`ps -ef`
All processes (BSD)	`ps aux`
Process tree	`ps -ejH` or `ps axjf`
Threads	`ps -eLf` or `ps axms`
Security info	`ps -eo euser,ruser,...`
Root-owned processes	`ps -U root -u root u`
Custom format	`ps -eo pid,comm,%cpu,%mem`
Process ID only	`ps -C <proc-name> -o pid=`
Process name from PID	`ps -q <pid> -o comm=`

FINAL TIPS

- Use `ps --help` to explore all available fields and formatting options.
- Combine `ps`, `grep`, `awk`, `xargs`, and `kill` for powerful scripting.
- Use `pgrep` and `pkill` for process filtering and killing by name.
- Prefer `htop` or `top` for interactive/live monitoring.
- Use `ps -eo` in **cron jobs or logging agents** for periodic snapshots.