

Linux Process Management — Lab Report

Objective

Understanding and implementing process management concepts in Linux through hands-on examples.

Theory

◊ What is a Process?

A process is an instance of a running program with its own memory space and system resources.

◊ Process States

-  R (Running/Runnable)
-  S (Sleeping — waiting for event)
-  D (Uninterruptible sleep)
-  T (Stopped)
-  Z (Zombie)

◊ Process Types

-  Foreground: Connected to terminal
-  Background: Runs independently

◊ Process Identifiers

-  PID: Unique process ID
-  PPID: Parent process ID

◊ Core Concepts

-  Parent/Child processes
-  Zombie processes
-  Orphan processes

Commands & Examples

View all processes (detailed)

```
ps aux
```

Explanation: Shows all running processes with user, PID, %CPU, %MEM, VSZ, RSS, TTY, STAT, START, TIME, and COMMAND.

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	7.5	0.1	23580	14176	?	Ss	00:09	0:05	/sbin/init sp
root	2	0.1	0.0	0	0	?	S	00:09	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	S	00:09	0:00	[pool_workque]
root	4	0.0	0.0	0	0	?	I<	00:09	0:00	[kworker/R-rc]
root	5	0.0	0.0	0	0	?	I<	00:09	0:00	[kworker/R-sy]
root	6	0.0	0.0	0	0	?	I<	00:09	0:00	[kworker/R-kv]
root	7	0.0	0.0	0	0	?	I<	00:09	0:00	[kworker/R-sl]
root	8	0.0	0.0	0	0	?	I<	00:09	0:00	[kworker/R-ne]
root	9	0.0	0.0	0	0	?	I	00:09	0:00	[kworker/0:0-]
root	10	0.3	0.0	0	0	?	I	00:09	0:00	[kworker/0:1-]
root	11	0.0	0.0	0	0	?	I<	00:09	0:00	[kworker/0:0H]
root	12	0.0	0.0	0	0	?	I	00:09	0:00	[kworker/u20:]
root	13	0.0	0.0	0	0	?	I<	00:09	0:00	[kworker/R-mm]
root	14	0.0	0.0	0	0	?	I	00:09	0:00	[rcu_tasks_kt]
root	15	0.0	0.0	0	0	?	I	00:09	0:00	[rcu_tasks_ru]
root	16	0.0	0.0	0	0	?	I	00:09	0:00	[rcu_tasks_tr]
root	17	0.3	0.0	0	0	?	S	00:09	0:00	[ksoftirqd/0]
root	18	0.9	0.0	0	0	?	I	00:09	0:00	[rcu_preempt]
root	19	0.0	0.0	0	0	?	S	00:09	0:00	[rcu_exp_par_]
root	20	0.4	0.0	0	0	?	S	00:09	0:00	[rcu_exp_gp_k]
root	21	0.0	0.0	0	0	?	S	00:09	0:00	[migration/0]
root	22	0.0	0.0	0	0	?	S	00:09	0:00	[idle_inject/]
root	23	0.0	0.0	0	0	?	S	00:09	0:00	[cpuhp/0]
root	24	0.0	0.0	0	0	?	S	00:09	0:00	[cpuhp/1]
root	25	0.0	0.0	0	0	?	S	00:09	0:00	[idle_inject/]
root	26	2.1	0.0	0	0	?	S	00:09	0:01	[migration/1]
root	27	0.1	0.0	0	0	?	S	00:09	0:00	[ksoftirqd/1]
root	28	0.2	0.0	0	0	?	I	00:09	0:00	[kworker/1:0-]
root	29	0.0	0.0	0	0	?	I<	00:09	0:00	[kworker/1:0H]
root	30	0.0	0.0	0	0	?	S	00:09	0:00	[cpuhp/2]
root	31	0.0	0.0	0	0	?	S	00:09	0:00	[idle_inject/]

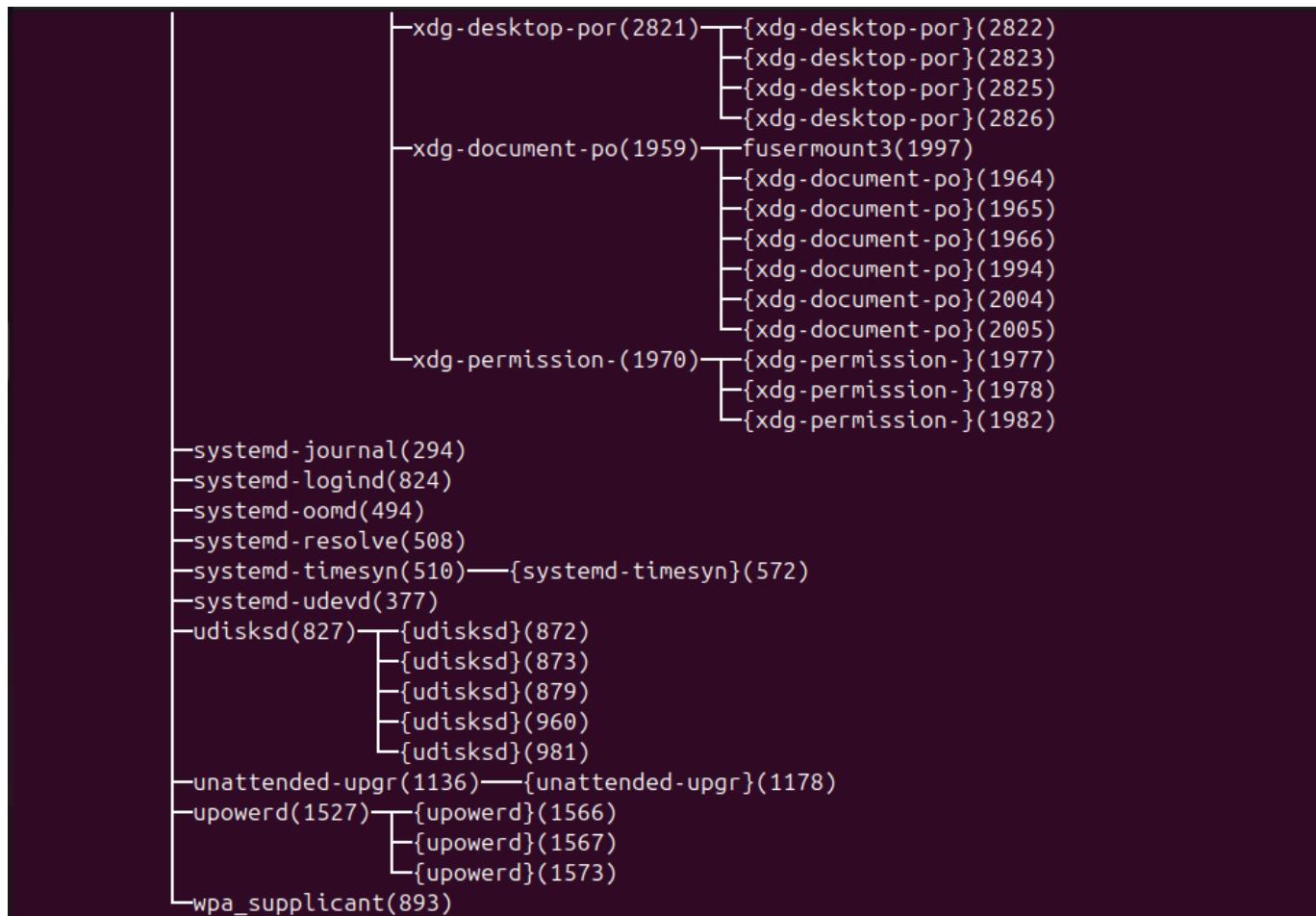
ujjwalt+	2414	2.2	0.1	471720	11164	?	Ssl	00:10	0:00	/usr/libexec/
ujjwalt+	2425	0.6	0.1	424904	14808	?	Sl	00:10	0:00	/usr/libexec/
ujjwalt+	2450	0.2	0.0	245312	7104	?	Sl	00:10	0:00	/usr/libexec/
ujjwalt+	2452	42.3	0.3	429636	27968	?	Sl	00:10	0:05	/usr/libexec/
ujjwalt+	2461	0.5	0.0	319092	7036	?	Sl	00:10	0:00	/usr/libexec/
ujjwalt+	2473	1.2	0.2	555328	23980	?	Sl	00:10	0:00	/usr/libexec/
ujjwalt+	2487	3.7	0.1	39136	11848	?	Ss	00:10	0:00	/snap/snapd-d
ujjwalt+	2493	0.3	0.0	318440	6208	?	Ssl	00:10	0:00	/usr/libexec/
ujjwalt+	2512	0.3	0.1	397792	8916	?	Sl	00:10	0:00	/usr/libexec/
ujjwalt+	2513	2.6	0.2	899384	24200	?	Ssl	00:10	0:00	/usr/libexec/
ujjwalt+	2522	0.4	0.0	318460	6200	?	Ssl	00:10	0:00	/usr/libexec/
ujjwalt+	2542	0.8	0.0	398044	7724	?	Ssl	00:10	0:00	/usr/libexec/
ujjwalt+	2566	0.6	0.0	319428	6744	?	Ssl	00:10	0:00	/usr/libexec/
ujjwalt+	2580	1.9	0.3	833976	29684	?	Ssl	00:10	0:00	/usr/libexec/
ujjwalt+	2603	5.3	0.3	429612	30568	?	Sl	00:10	0:00	/snap/snapd-d
ujjwalt+	2642	0.4	0.0	245436	7108	?	Sl	00:10	0:00	/usr/libexec/
ujjwalt+	2665	0.4	0.0	230220	5564	?	Ssl	00:10	0:00	/usr/libexec/
ujjwalt+	2674	6.7	0.6	946744	50312	?	Sl	00:10	0:00	/usr/bin/naut
ujjwalt+	2675	0.7	0.0	244936	6148	?	Ssl	00:10	0:00	/usr/libexec/
ujjwalt+	2676	1.4	0.1	618108	8884	?	Sl	00:10	0:00	/usr/libexec/
ujjwalt+	2709	2.6	0.1	562804	13816	?	Ssl	00:10	0:00	/usr/libexec/
root	2739	0.0	0.0	0	0	?	I<	00:10	0:00	[kworker/u27:]
root	2740	0.0	0.0	0	0	?	I<	00:10	0:00	[kworker/u27:]
ujjwalt+	2754	4.1	0.2	742932	24924	?	SNsl	00:10	0:00	/usr/libexec/
ujjwalt+	2755	5.4	0.4	925996	40952	?	Ssl	00:10	0:00	/usr/libexec/
ujjwalt+	2766	11.3	0.7	3019200	61592	?	Sl	00:10	0:00	gjs /usr/shar
ujjwalt+	2783	2.0	0.3	2544192	27792	?	Sl	00:10	0:00	/usr/bin/gjs
ujjwalt+	2821	4.8	0.2	425872	24388	?	Ssl	00:10	0:00	/usr/libexec/
ujjwalt+	2864	29.2	0.6	561652	51488	?	Ssl	00:10	0:01	/usr/libexec/
ujjwalt+	2871	1.6	0.0	19692	5056	pts/0	Ss	00:10	0:00	bash
ujjwalt+	2879	150	0.0	22284	4620	pts/0	R+	00:10	0:00	ps aux

Hierarchical process view

```
pstree -p
```

Explanation: Displays processes in a tree form with PIDs — useful to see parent/child relations.

```
ujjwaltyagi@ujjwaltyagi:~$ pstree -p
systemd(1)─ ModemManager(976)─ {ModemManager}(1006)
              └─ {ModemManager}(1013)
                  └─ {ModemManager}(1017)
              └─ NetworkManager(878)─ {NetworkManager}(978)
                  └─ {NetworkManager}(983)
                      └─ {NetworkManager}(984)
              └─ accounts-daemon(801)─ {accounts-daemon}(857)
                  └─ {accounts-daemon}(861)
                      └─ {accounts-daemon}(869)
              └─ avahi-daemon(768)─ avahi-daemon(880)
              └─ canonical-livep(1132)─ {canonical-livep}(1425)
                  └─ {canonical-livep}(1429)
                  └─ {canonical-livep}(1430)
                  └─ {canonical-livep}(1432)
                  └─ {canonical-livep}(1442)
                  └─ {canonical-livep}(1443)
                  └─ {canonical-livep}(1444)
                  └─ {canonical-livep}(1445)
                  └─ {canonical-livep}(1446)
                  └─ {canonical-livep}(1447)
              └─ colord(1480)─ {colord}(1485)
                  └─ {colord}(1486)
                      └─ {colord}(1488)
              └─ cron(804)
              └─ cups-browsed(1158)─ {cups-browsed}(1185)
                  └─ {cups-browsed}(1186)
                      └─ {cups-browsed}(1191)
              └─ cupsd(1128)─ dbus(1155)
              └─ dbus-daemon(769)
              └─ fwupd(4425)─ {fwupd}(4426)
                  └─ {fwupd}(4427)
                      └─ {fwupd}(4428)
```



Real-time monitoring

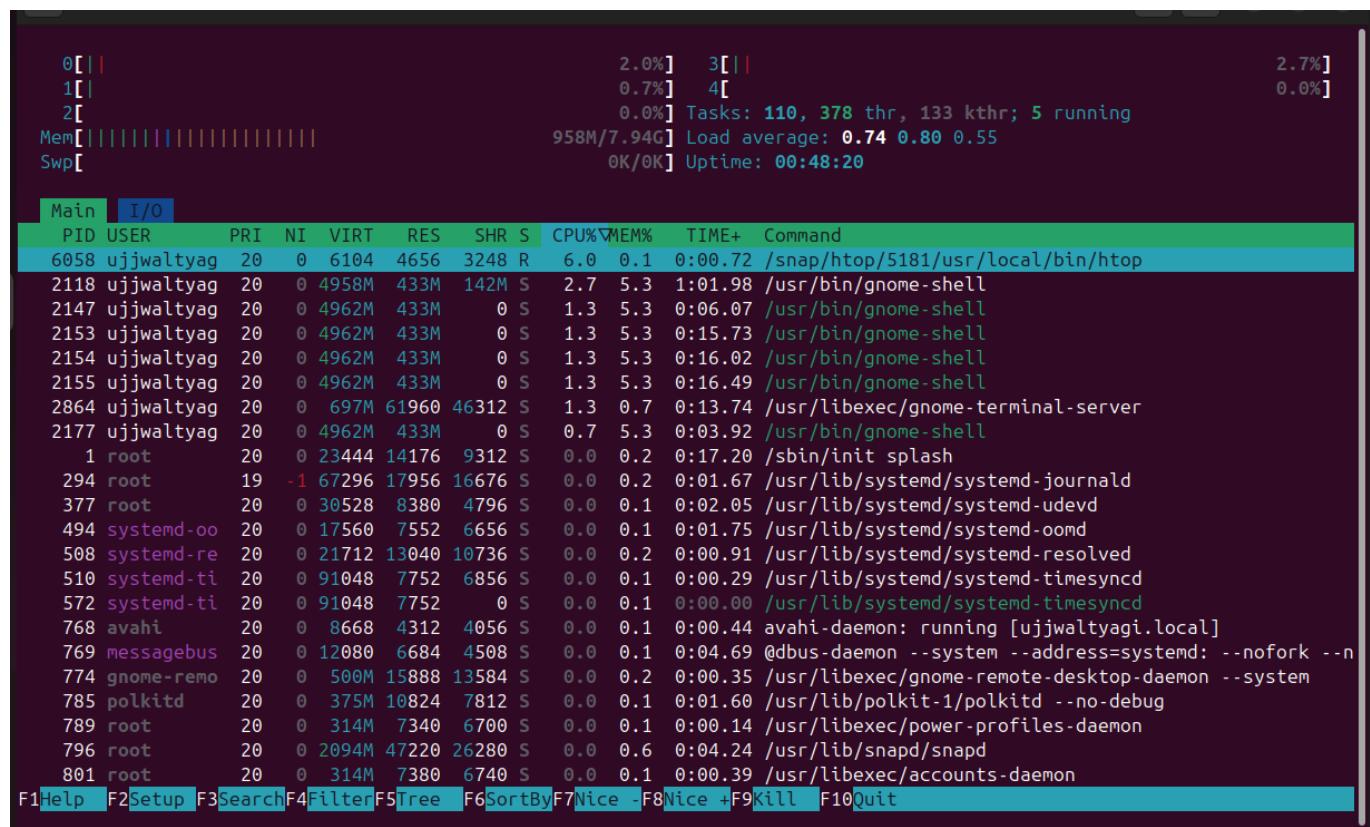
```
top
```

Explanation: Interactive display of processes sorted by CPU usage. Press **q** to quit.

top - 00:52:03 up 42 min, 1 user, load average: 0.06, 0.31, 0.37													
Tasks: 235 total, 1 running, 234 sleeping, 0 stopped, 0 zombie													
%Cpu(s): 0.2 us, 0.4 sy, 0.0 ni, 99.3 id, 0.0 wa, 0.0 hi, 0.1 si, 0.0 st													
MiB Mem : 8128.5 total, 5341.6 free, 1179.7 used, 1886.5 buff/cache													
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 6948.8 avail Mem													
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND		
2118	ujjwalt+	20	0	5077084	444020	145712	S	4.0	5.3	1:55.29	gnome-shell		
2864	ujjwalt+	20	0	638184	60264	45000	S	0.5	0.7	0:09.03	gnome-terminal-		
3525	root	20	0	0	0	0	I	0.3	0.0	0:00.54	kworker/u21:1-flush-8:0		
1	root	20	0	23444	14176	9312	S	0.0	0.2	0:10.56	systemd		
2	root	20	0	0	0	0	S	0.0	0.0	0:00.19	kthreadd		
3	root	20	0	0	0	0	S	0.0	0.0	0:00.00	pool_workqueue_release		
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-rcu_gp		
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-sync_wq		
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-kvfree_rcu_reclaim		
7	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-slab_flushwq		
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-netns		
10	root	20	0	0	0	0	I	0.0	0.0	0:05.36	kworker/0:1-events		
12	root	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/u20:0-ipv6_addrconf		
13	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/R-mm_percpu_wq		
14	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_kthread		
15	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_rude_kthread		
16	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_tasks_trace_kthread		
17	root	20	0	0	0	0	S	0.0	0.0	0:04.77	ksoftirqd/0		
18	root	20	0	0	0	0	I	0.0	0.0	0:03.84	rcu_preempt		
19	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_exp_par_gp_kthread_worker/0		
20	root	20	0	0	0	0	S	0.0	0.0	0:00.42	rcu_exp_gp_kthread_worker		
21	root	rt	0	0	0	0	S	0.0	0.0	0:00.12	migration/0		
22	root	-51	0	0	0	0	S	0.0	0.0	0:00.00	idle_inject/0		
23	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0		
24	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1		

htop

Explanation: Enhanced interactive viewer with process tree, colors, and easier navigation.



Run process in background

```
sleep 300 &
```

Explanation: Starts `sleep` in background; shell prints the job number and PID.

```
ujjwaltyagi@ujjwaltyagi:~$ sleep 300 &
[1] 6131
sleep 300 &
```

List jobs in current shell

```
jobs -l
```

Explanation: Lists jobs started from the current shell with job IDs and PIDs.

```
ujjwaltyagi@ujjwaltyagi:~$ jobs -l
[1]+ 6131 Running                  sleep 300 &
ujjwaltyagi@ujjwaltyagi:~$
```

Bring job to foreground / send to background

```
fg %1
bg %1
```

Explanation: `fg` brings job 1 to foreground; `bg` resumes job 1 in background.

```
ujjwaltyagi@ujjwaltyagi:~$ fg %1
sleep 300
```

Dismiss a background job

```
disown %1
```

Explanation: Removes job 1 from shell's job table so it will not receive SIGHUP on shell exit.

```
ujjwaltyagi@ujjwaltyagi:~$ sleep 400 &
[1] 3458
ujjwaltyagi@ujjwaltyagi:~$ jobs
[1]+  Running                  sleep 400 &
ujjwaltyagi@ujjwaltyagi:~$ disown %1
ujjwaltyagi@ujjwaltyagi:~$ jobs
ujjwaltyagi@ujjwaltyagi:~$ █
```

Run detached (nohup)

```
nohup python3 server.py > server.log 2>&1 &
```

Explanation: Starts `server.py` immune to hangups; output redirected to `server.log`.

```
ujjwaltyagi@ujjwaltyagi:~$ nohup python3 server.py > server.log 2>1 &
[1] 7049
ujjwaltyagi@ujjwaltyagi:~$ █
```

Find process by name / PID

```
pgrep -a sshd
pidof bash
```

Explanation: `pgrep -a` lists PIDs and command lines matching `sshd`. `pidof` returns PID(s) for a program.

```
ujjwaltyagi@ujjwaltyagi:~$ pgrep -a sshd
[1]+  Exit 2                  nohup python3 server.py > server.log 2> 1
ujjwaltyagi@ujjwaltyagi:~$ pidof bash
6260
ujjwaltyagi@ujjwaltyagi:~$ █
```

Kill a process (graceful then force)

```
kill PID                      # sends SIGTERM by default
kill -9 PID                    # sends SIGKILL (force)
pkill -f processname
```

Explanation: `kill` sends signals to PIDs; `pkill -f` matches the full command line.

```
ujjwaltyagi@ujjwaltyagi:~$ sleep 1200 &
[1] 9184
ujjwaltyagi@ujjwaltyagi:~$ ps
\ PID TTY          TIME CMD
 6260 pts/0    00:00:00 bash
 9184 pts/0    00:00:00 sleep
 9277 pts/0    00:00:00 ps
ujjwaltyagi@ujjwaltyagi:~$ ps aux | grep sleep
ujjwalt+  9184  0.0  0.0  16956  1908 pts/0      S     01:17   0:00 sleep 1200
ujjwalt+  9524  0.0  0.0  17812  2288 pts/0      S+    01:18   0:00 grep --color=
auto sleep
ujjwaltyagi@ujjwaltyagi:~$ kill 9184
ujjwaltyagi@ujjwaltyagi:~$ ps aux | grep sleep
ujjwalt+  9688  0.0  0.0  17812  2272 pts/0      S+    01:19   0:00 grep --color=
auto sleep
[1]+  Terminated                  sleep 1200
```

```
ujjwaltyagi@ujjwaltyagi:~$ sleep 1345 &
[1] 10099
ujjwaltyagi@ujjwaltyagi:~$ ps aux | grep sleep
ujjwalt+  10099  0.0  0.0  16956  1980 pts/0      S     01:20   0:00 sleep 1345
ujjwalt+  10122  0.0  0.0  17812  2284 pts/0      S+    01:20   0:00 grep --color=
auto sleep
ujjwaltyagi@ujjwaltyagi:~$ kill -9 10099
ujjwaltyagi@ujjwaltyagi:~$ ps aux | grep sleep
ujjwalt+  10208  0.0  0.0  17812  2288 pts/0      S+    01:21   0:00 grep --color=
auto sleep
[1]+  Killed                      sleep 1345
```

```
ujjwaltyagi@ujjwaltyagi:~$ sleep 2312 &
[2] 10704
ujjwaltyagi@ujjwaltyagi:~$ ps aux | grep sleep
ujjwalt+  10393  0.0  0.0  16956  1980 pts/0      S+    01:22   0:00 sleep 2345
ujjwalt+  10542  0.0  0.0  16956  1980 pts/1      S     01:22   0:00 sleep 2312
ujjwalt+  10704  0.0  0.0  16956  1980 pts/1      S     01:23   0:00 sleep 2312
ujjwalt+  10739  0.0  0.0  17812  2288 pts/1      S+    01:23   0:00 grep --color=
auto sleep
ujjwaltyagi@ujjwaltyagi:~$ pkill -f 10393
ujjwaltyagi@ujjwaltyagi:~$ ps aux | grep sleep
ujjwalt+  10393  0.0  0.0  16956  1980 pts/0      S+    01:22   0:00 sleep 2345
ujjwalt+  10542  0.0  0.0  16956  1980 pts/1      S     01:22   0:00 sleep 2312
ujjwalt+  10704  0.0  0.0  16956  1980 pts/1      S     01:23   0:00 sleep 2312
ujjwalt+  10889  0.0  0.0  17812  2288 pts/1      S+    01:23   0:00 grep --color=
auto sleep
```

Adjust process priority at start

```
nice -n 10 ./compute-heavy.sh
```

Explanation: Launches `compute-heavy.sh` with niceness 10 (lower priority).

```
ujjwaltyagi@ujjwaltyagi:~$ nice -n 10 ./gcd_lcm.sh 34 56 &
[1] 3511
ujjwaltyagi@ujjwaltyagi:~$ GCD(34,56) = 2
LCM(34,56) = 952
■
```

Change priority of running process

```
renice +5 -p 12345
renice -n -10 -p 12345
renice -n 5 -p 12345
```

Explanation: `renice` changes niceness of PID 12345 (positive increases nice value = lower priority).

```
ujjwaltyagi@ujjwaltyagi:~$ nice -n 10 sleep 300 &
[2] 3531
ujjwaltyagi@ujjwaltyagi:~$ sudo renice -n 5 -p 3531
[sudo] password for ujjwaltyagi:
3531 (process ID) old priority 10, new priority 5
ujjwaltyagi@ujjwaltyagi:~$ ■
```

View process open files and resources

```
lsof -p 12345
```

Explanation: Lists files, sockets, and resources opened by PID 12345.

```
ujjwaltyagi@ujjwaltyagi:~$ sleep 450 &
[4] 3540
ujjwaltyagi@ujjwaltyagi:~$ lsof -p 3540
COMMAND  PID   USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sleep  3540 ujjwaltyagi cwd   DIR    8,2      4096  920385 /home/ujjwaltyagi
sleep  3540 ujjwaltyagi rtd   DIR    8,2      4096     2 /
sleep  3540 ujjwaltyagi txt   REG    8,2     35336  525644 /usr/bin/sleep
sleep  3540 ujjwaltyagi mem   REG    8,2 14596880  526973 /usr/lib/locale/loc
le-archive
sleep  3540 ujjwaltyagi mem   REG    8,2   2125328  535783 /usr/lib/x86_64-linu
x-gnu/libc.so.6
sleep  3540 ujjwaltyagi mem   REG    8,2     613  567516 /usr/share/locale-la
ngpack/en/LC_MESSAGES/coreutils.mo
sleep  3540 ujjwaltyagi mem   REG    8,2   236616  535601 /usr/lib/x86_64-linu
x-gnu/ld-linux-x86-64.so.2
sleep  3540 ujjwaltyagi  0u   CHR   136,1      0t0      4 /dev/pts/1
sleep  3540 ujjwaltyagi  1u   CHR   136,1      0t0      4 /dev/pts/1
sleep  3540 ujjwaltyagi  2u   CHR   136,1      0t0      4 /dev/pts/1
uijwaltyagi@uijwaltyagi:~$
```

Show process tree for a PID

```
ps f -p 12345
```

Explanation: Shows process hierarchy for PID 1 (or change PID as needed).

```
ujjwaltyagi@ujjwaltyagi:~$ sleep 700 &
[1] 3558
ujjwaltyagi@ujjwaltyagi:~$ ps f -p 3558
  PID TTY      STAT   TIME COMMAND
 3558 pts/0    S      0:00 sleep 700
ujjwaltyagi@ujjwaltyagi:~$
```

⌚ Advanced Topics & Tools

This section adds detailed theory, commands and short examples for advanced process-management topics that supplement the previous sections.

Signals (SIGTERM vs SIGKILL and common signals)

- SIGTERM (15) — polite request to terminate; process can catch and clean up.
- SIGKILL (9) — immediate forced termination; cannot be caught or ignored.
- SIGHUP (1) — hangup; commonly used to tell daemons to reload config.
- SIGSTOP / SIGCONT — stop and continue a process (job control).

Example:

```
# polite terminate  
kill -15 3050  
  
# force kill  
kill -9 3050
```

Explanation: Prefer SIGTERM then SIGKILL only if process doesn't exit.

```
ujjwaltyagi@ujjwaltyagi:~$ sleep 3050 &  
[2] 3763  
ujjwaltyagi@ujjwaltyagi:~$ echo $!  
3763  
ujjwaltyagi@ujjwaltyagi:~$ kill -15 3763  
ujjwaltyagi@ujjwaltyagi:~$ ps -p 3763  
    PID TTY          TIME CMD  
[2]+  Terminated            sleep 3050  
ujjwaltyagi@ujjwaltyagi:~$ sleep 3050 &  
[2] 4550  
ujjwaltyagi@ujjwaltyagi:~$ kill -9 4550  
ujjwaltyagi@ujjwaltyagi:~$ ps -p 4550  
    PID TTY          TIME CMD  
[2]+  Killed             sleep 3050  
ujjwaltyagi@ujjwaltyagi:~$
```

CPU Affinity — taskset (bind process to CPU cores)

- Use CPU affinity to bind a process to specific CPU core(s).

```
# view current affinity  
taskset -cp 3050  
  
# set affinity to core 1 only  
taskset -cp 1 3050  
  
# start a process on cores 0-1  
taskset -c 0,1 ./compute-heavy.sh
```

Explanation: Useful to reduce cache thrashing or isolate workloads.

```
ujjwaltyagi@ujjwaltyagi:~$ sleep 3050 &
[2] 4553
ujjwaltyagi@ujjwaltyagi:~$ taskset -cp 4553
pid 4553's current affinity list: 0-4
ujjwaltyagi@ujjwaltyagi:~$ sleep 3050 &
[3] 4566
ujjwaltyagi@ujjwaltyagi:~$ taskset -cp 1 4566
pid 4566's current affinity list: 0-4
pid 4566's new affinity list: 1
ujjwaltyagi@ujjwaltyagi:~$ sleep 3050 &
[4] 4568
ujjwaltyagi@ujjwaltyagi:~$ taskset -c 0,1 ./print-number.sh
1
2
3
4
5
6
7
```

I/O Scheduling Priority — ionice

- Controls disk I/O scheduling class and priority.

```
# set process to idle I/O class (only uses disk when idle)
ionice -c 3 -p 3050

# run a command with best-effort class and priority 7
ionice -c 2 -n 7 tar -czf backup.tar.gz /home
```

Explanation: Helps limit I/O interference from background jobs.

```
ujjwaltyagi@ujjwaltyagi:~$ sleep 3400 &
[2] 4730
ujjwaltyagi@ujjwaltyagi:~$ sudo ionice -c 3 -p 4730
ujjwaltyagi@ujjwaltyagi:~$ ionice -p 4730
idle
```

```
ujjwaltyagi@ujjwaltyagi:~$ mkdir testdir
ujjwaltyagi@ujjwaltyagi:~$ echo "hello world" > testdir/file1.txt
ujjwaltyagi@ujjwaltyagi:~$ echo "more data" > testdir/file2.txt
ujjwaltyagi@ujjwaltyagi:~$ ionice -c 2 -n 7 tar -czvf backup.tar.gz testdir/
testdir/
testdir/file2.txt
testdir/file1.txt
ujjwaltyagi@ujjwaltyagi:~$
```

Per-Process Statistics — pidstat

- Monitor per-process CPU usage over time.

```
# sample PID 3050 every 2 seconds, 3 samples
pidstat -p 3050 2 3
```

Explanation: Useful to observe CPU trends for a specific PID.

```
ujjwaltyagi@ujjwaltyagi:~$ dd if=/dev/zero of=testfile bs=1M count=5000 &
[1] 4847
ujjwaltyagi@ujjwaltyagi:~$ pid=$!
ujjwaltyagi@ujjwaltyagi:~$ sudo ionice -c 3 -p $pid
ujjwaltyagi@ujjwaltyagi:~$ pidstat -p $pid 2 5
Linux 6.14.0-29-generic (ujjwaltyagi)   11/01/2025      _x86_64_          (5 CPU)

09:39:00 PM    UID      PID    %usr %system  %guest    %wait    %CPU     CPU  Command
09:39:02 PM  1000    4847    0.00  99.50    0.00    0.00  99.50      3  dd
09:39:04 PM  1000    4847    0.00 100.50    0.00    0.00 100.50      3  dd
09:39:06 PM  1000    4847    0.42  99.58    0.00    0.00 100.00      3  dd
09:39:08 PM  1000    4847    0.00  97.51    0.00    2.49  97.51      1  dd
09:39:10 PM  1000    4847    0.00  98.51    0.00    1.49  98.51      1  dd
Average:    1000    4847    0.10  99.14    0.00    0.77  99.23      -  dd
ujjwaltyagi@ujjwaltyagi:~$ 5000+0 records in
5000+0 records out
5242880000 bytes (5.2 GB, 4.9 GiB) copied, 46.481 s, 113 MB/s
```

Trace System Calls — strace

- Attach to a running process and inspect system calls.

```
# attach and view syscalls
sudo strace -p 3050

# run a command under strace
strace -o trace.txt ls
```

Explanation: Great for debugging why a process is failing (file access, network calls).

```
ujjwaltyagi@ujjwaltyagi:~$ sleep 300 &
[1] 4906
ujjwaltyagi@ujjwaltyagi:~$ sudo strace -p 4906
[sudo] password for ujjwaltyagi:
strace: Process 4906 attached
restart_syscall(<... resuming interrupted read ...>)
```

```
ujjwaltyagi@ujjwaltyagi:~$ sudo strace -o trace.txt ls
1           areaper          global
1_10_1.c    area_perimeter.c int
1_10_2.c    backup           int.c
1_10_3.c    backup.tar.gz   kmtom
1_10_4.c    block            largest
12          calc.c           largest_smallest
12.c        calculator       largest_smallest.c
1.c         calculator.c    largest_smallest_macro
2           checkfile1.sh    largest_smallest_macro.c
25596_1.c   check_file_permission.sh local
25596_1.c   checkfile.sh    marks
25596_1.txt count           max3
25596_2.c   count_lines_words.sh maximum_of_three_num.c
25596_2.c   count-line-words.sh modify
25596_3.c   'c programs'   modify1
25596_4_2.c Desktop          modify1.c
25596_4.c   dimen           modify1.txt
25596_4.txt Documents        modify.c
29_09       Downloads        Music
2int        ds_starter       myproject
3           enhanced_number.sh palindrome.sh
30_09_1.c   evenodd          Pictures
30_09_2.c   even_odd.c      print-number.sh
30_09_3.c   factorial_function.sh print_number.sh
30_09_4.c   factorial-fun.sh product
30_09_5.c   floatint.c      project6s
30_09_6.c   floatint.txt    projects
app.py      gcd_lcm.sh      Public
ujjwaltyagi@ujjwaltyagi:~$
```

Find Process Using a Port — fuser / ss / netstat

```
# find PID using TCP port 8080
sudo fuser -n tcp 8080

# alternative: list sockets and PIDs
sudo ss -tuln | grep 8080
```

Explanation: Identifies which process is bound to a port (useful before killing or reconfiguring).

```
ujjwaltyagi@ujjwaltyagi:~$ sudo fuser -n tcp 8080
[sudo] password for ujjwaltyagi:
8080/tcp:          5153
ujjwaltyagi@ujjwaltyagi:~$ sudo ss -tuln -n tcp 8080
Error: an inet prefix is expected rather than "tcp".
Cannot parse dst/src address.
ujjwaltyagi@ujjwaltyagi:~$ sudo fuser -n tcp 8080
8080/tcp:          5153
ujjwaltyagi@ujjwaltyagi:~$ sudo ss -tuln | grep 8080
tcp    LISTEN  0      5                  0.0.0.0:8080          0.0.0.0:*
ujjwaltyagi@ujjwaltyagi:~$
```

Control Groups (cgroups) — basic resource limiting

- Create a cgroup, set CPU and memory limits, add a PID:

```
# create cgroup (requires cgroup-tools or systemd)
sudo cgcreate -g cpu,memory:/testgroup

# limit CPU (cfs quota in microseconds)
echo 50000 | sudo tee /sys/fs/cgroup/cpu/testgroup/cpu.cfs_quota_us

# limit memory to 100MB
echo $((100*1024*1024)) | sudo tee
/sys/fs/cgroup/memory/testgroup/memory.limit_in_bytes

# add PID 3050 to the cgroup
echo 3050 | sudo tee /sys/fs/cgroup/cpu/testgroup/cgroup.procs
```

Explanation: cgroups provide strong isolation and resource controls for processes or groups.

```
ujjwaltyagi@ujjwaltyagi:~$ sudo cgcreate -g cpu,memory:/testgroup
ujjwaltyagi@ujjwaltyagi:~$ echo 50000 | sudo tee /sys/fs/cgroup/cpu/testgroup/cpu.cfs_quota_us
tee: /sys/fs/cgroup/cpu/testgroup/cpu.cfs_quota_us: No such file or directory
50000
ujjwaltyagi@ujjwaltyagi:~$ echo 50000 | sudo tee /sys/fs/cgroup/cpu/testgroup/cpu.cfs_quota_us
tee: /sys/fs/cgroup/cpu/testgroup/cpu.cfs_quota_us: No such file or directory
50000
ujjwaltyagi@ujjwaltyagi:~$ sudo cgcreate -g cpu,memory:/testgroup
ujjwaltyagi@ujjwaltyagi:~$ sudo cgcreate -g cpu,memory:/testgroup
ujjwaltyagi@ujjwaltyagi:~$ echo 50000 | sudo tee /sys/fs/cgroup/cpu/testgroup/cpu.cfs_quota_us
tee: /sys/fs/cgroup/cpu/testgroup/cpu.cfs_quota_us: No such file or directory
50000
ujjwaltyagi@ujjwaltyagi:~$ sudo mkdir /sys/fs/cgroup/testgroup
mkdir: cannot create directory '/sys/fs/cgroup/testgroup': File exists
ujjwaltyagi@ujjwaltyagi:~$ echo "50000 100000" | sudo tee /sys/fs/cgroup/testgroup/cpu.max
50000 100000
ujjwaltyagi@ujjwaltyagi:~$ echo $((100*1024*1024)) | sudo tee /sys/fs/cgroup/testgroup/cpu.max
104857600
ujjwaltyagi@ujjwaltyagi:~$ sleep 3050 &
[1] 10488
ujjwaltyagi@ujjwaltyagi:~$ echo 10488 | sudo tee /sys/fs/cgroup/testgroup/cpu.procs
tee: /sys/fs/cgroup/testgroup/cpu.procs: Permission denied
10488
ujjwaltyagi@ujjwaltyagi:~$ echo 10488 | sudo tee /sys/fs/cgroup/testgroup/cgroup.procs
10488
ujjwaltyagi@ujjwaltyagi:~$ echo $$ | sudo tee /sys/fs/cgroup/testgroup/cgroup.procs
8473
ujjwaltyagi@ujjwaltyagi:~$ cat /sys/fs/cgroup/testgroup/cgroup.procs
10488
8473
10505
ujjwaltyagi@ujjwaltyagi:~$
```

Real-Time Scheduling — chrt

- Set real-time scheduling policies (FIFO or RR).

```
# run command with FIFO policy, priority 50
sudo chrt -f 50 sleep 1000

# change scheduling of running PID
sudo chrt -p 50 3050
```

Explanation: Use with caution — real-time tasks can starve normal processes.

```
ujjwaltyagi@ujjwaltyagi:~$ sudo chrt -f 50 sleep 60 & echo $!
[2] 7052
7052
ujjwaltyagi@ujjwaltyagi:~$ [sudo] password for ujjwaltyagi:
2911019789
2911019789: command not found

[2]+  Stopped                  sudo chrt -f 50 sleep 60
```

```
ujjwaltyagi@ujjwaltyagi:~$ sleep 3050 &
[6] 7116
ujjwaltyagi@ujjwaltyagi:~$ sudo chrt -p 50 7116
ujjwaltyagi@ujjwaltyagi:~$ s
```

schedtool (advanced scheduling tweaks)

```
# set round-robin priority 10 on PID
sudo schedtool -R -p 10 3050
```

Explanation: Alternative low-level scheduling tweaks; platform-dependent.

```
ujjwaltyagi@ujjwaltyagi:~$ sleep 200 &
[2] 8259
ujjwaltyagi@ujjwaltyagi:~$ sudo schedtool -R -p 10 8259
ujjwaltyagi@ujjwaltyagi:~$ schedtool 8259
PID 8259: PRIO 10, POLICY R: SCHED_RR      , NICE 0, AFFINITY 0x1f
ujjwaltyagi@ujjwaltyagi:~$
```

systemd-run — temporary scopes with resource settings

```
# run stress under systemd scope with CPUWeight control
systemd-run --scope -p CPUWeight=200 stress --cpu 4

# run a command and limit memory
systemd-run --scope -p MemoryMax=200M ./compute-heavy.sh
```

Explanation: Convenient to run one-off tasks with cgroup-based limits via systemd.

```
ujjwaltyagi@ujjwaltyagi:~$ sudo systemd-run --scope -p CPUWEIGHT=200
Command line to execute required.
ujjwaltyagi@ujjwaltyagi:~$ stress --cpu 4
stress: info: [8464] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
```

```
ujjwaltyagi@ujjwaltyagi:~$ nano compute-heavy.sh
ujjwaltyagi@ujjwaltyagi:~$ nano compute-heavy.sh
ujjwaltyagi@ujjwaltyagi:~$ chmod 777 compute-heavy.sh
ujjwaltyagi@ujjwaltyagi:~$ sudo systemd-run --scope -p MemoryMax=200M ./compute-heavy.sh
Running as unit: run-r4f4ca10987434f0d880ab88581b5684d.scope; invocation ID: 155c8109066a424cb0c1d8e5c16a7c7f
stress: info: [8545] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
stress: info: [8545] successful run completed in 31s
ujjwaltyagi@ujjwaltyagi:~$
```

lsof — list open files/resources per process

```
lsof -p 3050 | head -10
```

Explanation: Shows files, sockets and devices a process has open; helpful for debugging leaks.

```
ujjwaltyagi@ujjwaltyagi:~$ sleep 300 &
[2] 10369
ujjwaltyagi@ujjwaltyagi:~$ lsof -p 10369 | head -10
COMMAND   PID   USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sleep  10369 ujjwaltyagi cwd   DIR    8,2     4096 920385 /home/ujjwaltyagi
sleep  10369 ujjwaltyagi rtd   DIR    8,2     4096      2 /
sleep  10369 ujjwaltyagi txt   REG    8,2    35336 525644 /usr/bin/sleep
sleep  10369 ujjwaltyagi mem   REG    8,2 14596880 526973 /usr/lib/locale/locale-archive
sleep  10369 ujjwaltyagi mem   REG    8,2 2125328 535783 /usr/lib/x86_64-linux-gnu/libc.so.6
sleep  10369 ujjwaltyagi mem   REG    8,2      613 567516 /usr/share/locale-langpack/en/LC_MESSAGES/coreutils.mo
sleep  10369 ujjwaltyagi mem   REG    8,2 236616 535601 /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
sleep  10369 ujjwaltyagi  0u   CHR   136,0     0t0      3 /dev/pts/0
sleep  10369 ujjwaltyagi  1u   CHR   136,0     0t0      3 /dev/pts/0
ujjwaltyagi@ujjwaltyagi:~$
```

Summary: Tools & When to Use Them

Tool	Purpose	Use When...
ps, top, htop	Inspect processes	Quick snapshot or live monitoring
pstree, ps f	Parent/child view	Understanding process hierarchy
kill, pkill	Send signals	Terminate or signal processes
nice, renice	CPU niceness	Lower priority background jobs
taskset	CPU affinity	Bind to specific cores
ionice	I/O priority	Reduce disk contention
pidstat	Per-process stats	Time-series CPU analysis

Tool	Purpose	Use When...
strace	Syscall tracing	Debugging failing processes
fuser, ss	Port/process mapping	Identify service owners of ports
cgroups, systemd-run	Resource limits	Enforce QoS and isolation
chrt, schedtool	Real-time scheduling	Real-time workloads (careful)

Practical examples (quick recap)

```
# Start low-priority background job
nice -n 10 sleep 300 &

# Bind a running PID to CPU 1
taskset -cp 1 3050

# Make process IO idle
ionice -c 3 -p 3050

# Trace syscalls
sudo strace -p 3050

# Limit memory with systemd-run
systemd-run --scope -p MemoryMax=150M sleep 600
```

Conclusion

This practical lab demonstrated essential Linux process management concepts through hands-on examples.

References & Resources

-  man pages: ps, top, kill, nice, renice
-  Linux kernel documentation
-  System administration guides

~ End of Lab Report ~