

Lms_Linux / lab7.md

rish4name changes c9916c3 · now

288 lines (179 loc) • 6.05 KB

🧠 Linux Process Management — Technical Assignment

A Guided Exploration of How Linux Really Thinks

◆ 1. Enumerating All Active Processes

🔧 Command

ps aux

✳️ Conceptual Insight

This classical invocation of `ps` provides a **snapshot of the entire process space**.

- `a` → Display processes of *all* users
- `u` → Include the effective user, CPU%, memory%, etc.
- `x` → Show processes detached from any controlling terminal

Together, they approximate a low-level “process census,” useful for diagnostic baselining.

📸 Output

```
rishabh-pandey@rishabh-pandey-VirtualBox: ~ ps aux
USER     PID %CPU %MEM    VSZ   RSS TTY      STAT START  TIME COMMAND
root      1  5.5  0.7 23456 14192 ?        Ss   18:40  0:02 /sbin/init splash
root      2  0.0  0.0      0   0 ?        S     18:40  0:00 [kthreadd]
root      3  0.0  0.0      0   0 ?        S     18:40  0:00 [pool_workqueue_release]
root      4  0.0  0.0      0   0 ?        I<   18:40  0:00 [kworker/R-rcu_gp]
root      5  0.0  0.0      0   0 ?        I<   18:40  0:00 [kworker/R-sync_wq]
root      6  0.0  0.0      0   0 ?        I<   18:40  0:00 [kworker/R-kvfree_rcu_reclaim]
root      7  0.0  0.0      0   0 ?        I<   18:40  0:00 [kworker/R-slab_flushwq]
root      8  0.0  0.0      0   0 ?        I<   18:40  0:00 [kworker/R-netns]
root      9  0.0  0.0      0   0 ?        I     18:40  0:00 [kworker/0:0-events]
root     10  0.3  0.0      0   0 ?        I     18:40  0:00 [kworker/0:1-ata_sff]
root     11  0.0  0.0      0   0 ?        I<   18:40  0:00 [kworker/0:0H-events_highpri]
root     12  0.0  0.0      0   0 ?        I     18:40  0:00 [kworker/u24:0 ipv6_addrconf]
root     13  0.0  0.0      0   0 ?        I<   18:40  0:00 [kworker/R-mm_percpu_wq]
root     14  0.0  0.0      0   0 ?        I     18:40  0:00 [rcu_tasks_kthread]
root     15  0.0  0.0      0   0 ?        I     18:40  0:00 [rcu_tasks_rude_kthread]
root     16  0.0  0.0      0   0 ?        I     18:40  0:00 [rcu_tasks_trace_kthread]
root     17  0.0  0.0      0   0 ?        S     18:40  0:00 [ksoftirqd/0]
root     18  0.6  0.0      0   0 ?        I     18:40  0:00 [rcu_preempt]
root     19  0.0  0.0      0   0 ?        S     18:40  0:00 [rcu_exp_par_gp_kthread_worker/0]
root     20  1.0  0.0      0   0 ?        S     18:40  0:00 [rcu_exp_gp_kthread_worker]
root     21  0.0  0.0      0   0 ?        S     18:40  0:00 [migration/0]
root     22  0.0  0.0      0   0 ?        S     18:40  0:00 [idle_inject/0]
root     23  0.0  0.0      0   0 ?        S     18:40  0:00 [cpuhp/0]
root     24  0.0  0.0      0   0 ?        S     18:40  0:00 [cpuhp/1]
root     25  0.0  0.0      0   0 ?        S     18:40  0:00 [idle_inject/1]
root     26  0.3  0.0      0   0 ?        S     18:40  0:00 [migration/1]
root     27  0.1  0.0      0   0 ?        S     18:40  0:00 [ksoftirqd/1]
root     28  0.0  0.0      0   0 ?        I     18:40  0:00 [kworker/1:0-events]
root     29  0.0  0.0      0   0 ?        I<   18:40  0:00 [kworker/1:0H-events_highpri]
root     30  0.0  0.0      0   0 ?        S     18:40  0:00 [cpuhp/2]
root     31  0.0  0.0      0   0 ?        S     18:40  0:00 [idle_inject/2]
root     32  0.4  0.0      0   0 ?        S     18:40  0:00 [migration/2]
root     33  0.0  0.0      0   0 ?        S     18:40  0:00 [ksoftirqd/2]
root     34  0.0  0.0      0   0 ?        T     18:40  0:00 [kworker/2:0-ccu_sp]
```

2. Visualizing the Process Hierarchy

Command

`pstree -p`



Conceptual Insight

Processes in Linux are not a flat list; they form a **hierarchical tree** rooted in `systemd` (or historically, `init`). `pstree` exposes these ancestry relations, making orphaned or runaway children immediately visible.

Output

```
rishabh-pandey@rishabh-pandey-VirtualBox: $ pstree -p
systemd(1)─{ModemManager}(992)─{ModemManager}(1026)
                         └─{ModemManager}(1033)
                         └─{ModemManager}(1035)
NetworkManager(852)─{NetworkManager}(951)
                         └─{NetworkManager}(953)
                         └─{NetworkManager}(957)
VBoxDRMClient(858)─{VBoxDRMClient}(878)
                         └─{VBoxDRMClient}(880)
                         └─{VBoxDRMClient}(884)
                         └─{VBoxDRMClient}(886)
                         └─{VBoxDRMClient}(2820)
VBoxService(863)─{VBoxService}(868)
                         └─{VBoxService}(870)
                         └─{VBoxService}(871)
                         └─{VBoxService}(872)
                         └─{VBoxService}(874)
                         └─{VBoxService}(877)
                         └─{VBoxService}(879)
                         └─{VBoxService}(883)
accounts-daemon(780)─{accounts-daemon}(866)
                         └─{accounts-daemon}(867)
                         └─{accounts-daemon}(873)
anacron(745)
avahi-daemon(748)─avahi-daemon(847)
colord(1435)─{colord}(1440)
                         └─{colord}(1441)
                         └─{colord}(1443)
cron(782)
cups-browsed(1188)─{cups-browsed}(1254)
                         └─{cups-browsed}(1255)
                         └─{cups-browsed}(1257)
cupsd(1145)
dbus-daemon(749)
gdm3(1169)─gdm-session-wor(1800)─gdm-wayland-ses(1924)─gnome-session-b(1929)─{gnome-session-b}(1983)
                         └─{gnome-session-b}(1984)

Nov 3 18:44
```



```
rishabh-pandey@rishabh-pandey-VirtualBox: ~
xdg-desktop-por(2742)─{xdg-desktop-por}(2763)
                         └─{xdg-desktop-por}(2764)
                         └─{xdg-desktop-por}(2765)
                         └─{xdg-desktop-por}(2796)
                         └─{xdg-desktop-por}(2823)
xdg-desktop-por(2841)─{xdg-desktop-por}(2843)
                         └─{xdg-desktop-por}(2844)
                         └─{xdg-desktop-por}(2846)
                         └─{xdg-desktop-por}(2848)
xdg-document-po(1892)─fusermount3(1903)
                         └─{xdg-document-po}(1893)
                         └─{xdg-document-po}(1894)
                         └─{xdg-document-po}(1895)
                         └─{xdg-document-po}(1902)
                         └─{xdg-document-po}(1905)
                         └─{xdg-document-po}(1906)
xdg-permission-(1896)─{xdg-permission-}(1897)
                         └─{xdg-permission-}(1898)
                         └─{xdg-permission-}(1900)
systemd-journal(295)
systemd-logind(794)
systemd-oomd(409)
systemd-resolve(411)
systemd-udevd(370)
udisksd(798)─{udisksd}(850)
                         └─{udisksd}(856)
                         └─{udisksd}(859)
                         └─{udisksd}(996)
                         └─{udisksd}(1007)
unattended-upgr(1153)─{unattended-upgr}(1196)
upowerd(1476)─{upowerd}(1512)
                         └─{upowerd}(1513)
                         └─{upowerd}(1514)
wpa_supplicant(855)

Nov 3 18:44
```



3. Observing Real-Time System Dynamics



Command

top



Conceptual Insight

`top` acts as a **living cross-section** of process activity — updated continuously. Use it to spot CPU hogs, memory leaks, or load spikes. ➡ Press `q` to exit before losing yourself in the numbers.

📸 Output

```

Tasks: 262 total, 1 running, 261 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.1 us, 0.3 sy, 0.0 ni, 99.6 id, 0.0 wa, 0.0 hi, 0.1 si, 0.0 st
MiB Mem : 1966.9 total, 107.3 free, 1177.5 used, 875.4 buff/cache
MiB Swap: 2048.0 total, 2047.8 free, 0.2 used. 789.4 avail Mem

PID USER PR NI VIRT RES SHR %CPU %MEM TIME+ COMMAND
2073 rishabh+ 20 0 4400436 296700 129640 S 3.0 14.7 0:33.71 gnome-shell
2940 rishabh+ 20 0 701128 57848 46152 S 1.3 2.9 0:03.69 gnome-terminal-
10 root 20 0 0 0 0 I 0.3 0.0 0:02.09 kworker/0:1-mm_percpu_wq
3279 root 20 0 0 0 0 I 0.3 0.0 0:00.31 kworker/5:0-events
1 root 20 0 23320 14192 9456 S 0.0 0.7 0:04.96 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.11 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-slub_flushwq
8 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/R-netsns
11 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:H-events_highpri
12 root 20 0 0 0 0 I 0.0 0.0 0:00.00 kworker/u24:0-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 0:00.10 ksoftirqd/0
18 root 20 0 0 0 0 I 0.0 0.0 0:02.85 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 0:00.61 rcu_exp_gp_kthread_worker
21 root rt 0 0 0 0 S 0.0 0.0 0:00.17 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
25 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/1
26 root rt 0 0 0 0 S 0.0 0.0 0:00.36 migration/1

```

⚡ 4. Manipulating Process Scheduling Priority

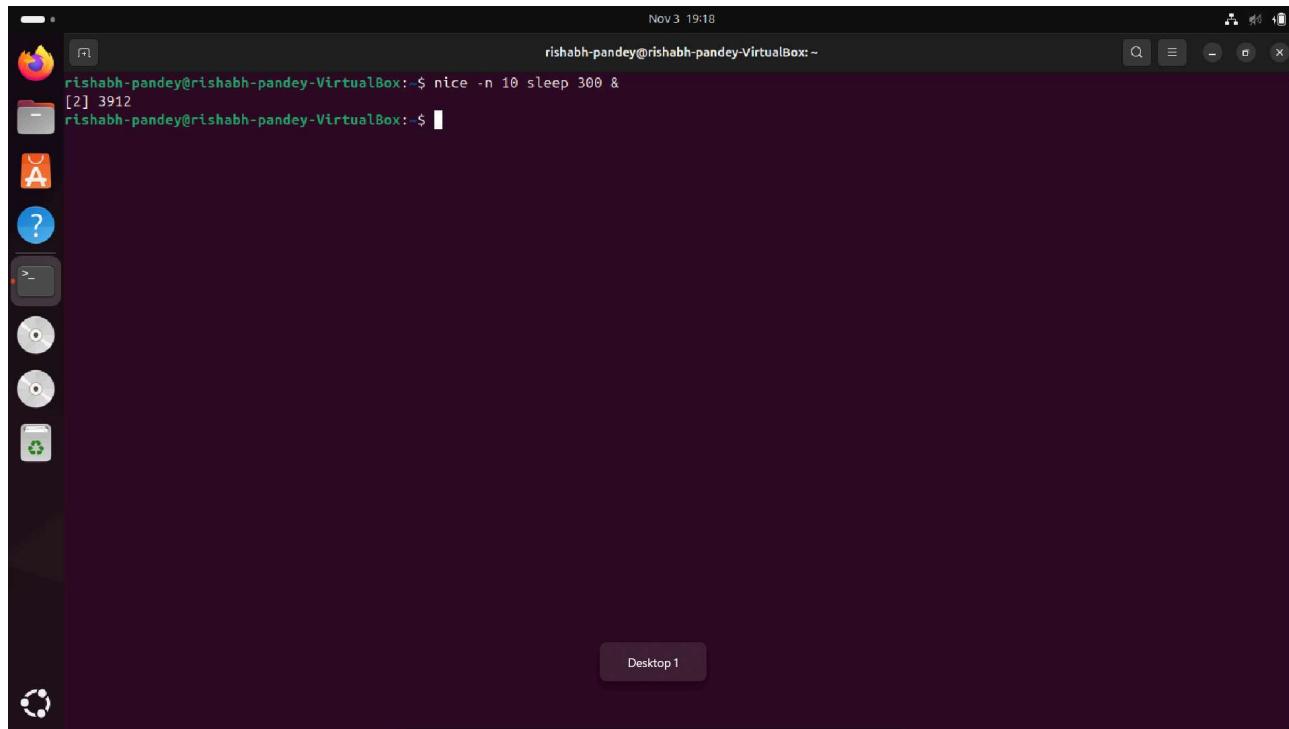
● Start a Process at Lower Priority

```
sudo nice -n 10 sleep 300 &
```



➡ Here, the process (PID 3050 assumed) is launched with a *niceness* of `10`, signaling lower scheduling priority.

📸 Output



A screenshot of a Linux desktop environment, likely Ubuntu, showing a terminal window. The terminal window has a dark background and contains the following text:

```
rishabh-pandey@rishabh-pandey-VirtualBox: ~ nice -n 10 sleep 300 &
[2] 3912
```

The terminal window is titled "rishabh-pandey@rishabh-pandey-VirtualBox: ~". The desktop interface includes a vertical dock on the left with icons for various applications like a browser, file manager, and system tools. A status bar at the top shows the date and time: "Nov 3 19:18". A small window titled "Desktop 1" is visible in the bottom right corner.

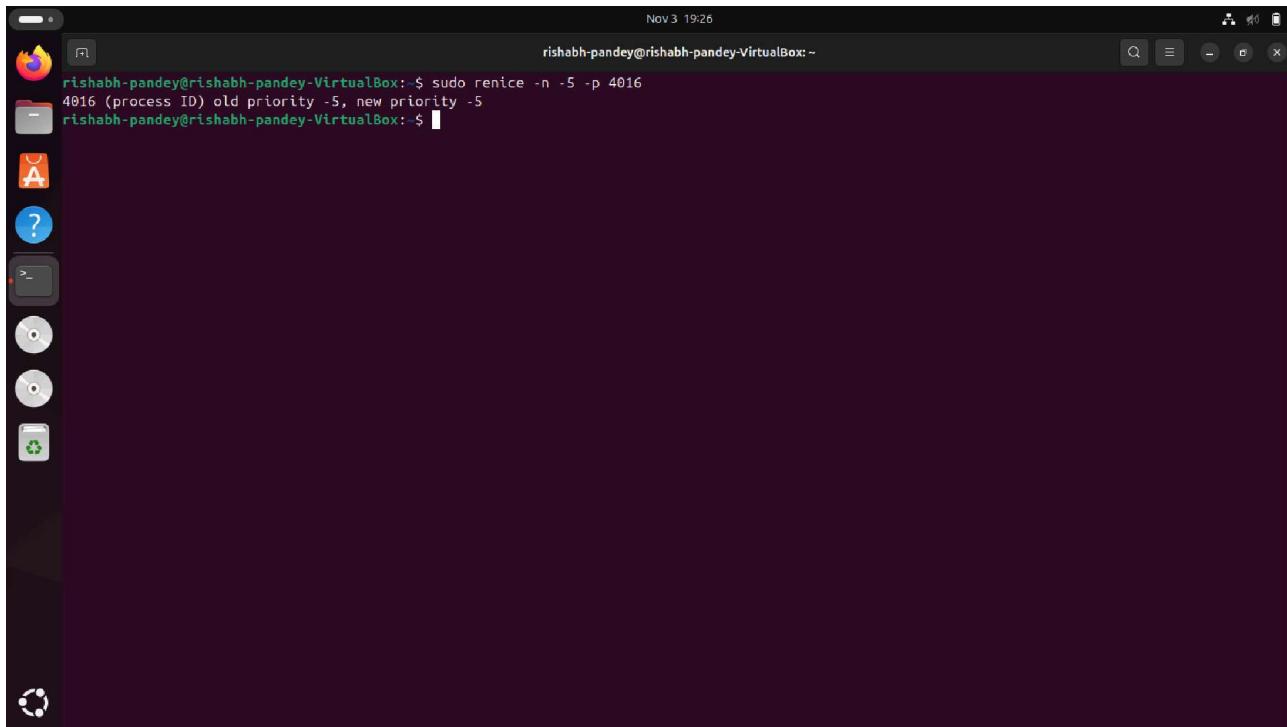
⌚ Reprioritize an Existing Process

```
renice -n -5 -p 3050
```



👉 Increasing priority (negative nice values) lets a process request more CPU attention — though it's still subject to kernel policy constraints.

📸 Output



A screenshot of a Linux desktop environment showing a terminal window. The terminal window has a dark background and contains the following text:

```
rishabh-pandey@rishabh-pandey-VirtualBox:~$ sudo renice -n -5 -p 4016
4016 (process ID) old priority -5, new priority -5
rishabh-pandey@rishabh-pandey-VirtualBox:~$
```

The terminal window is titled "Nov 3 19:26". The desktop interface includes a vertical dock on the left with icons for file manager, terminal, application switcher, help, and system status.

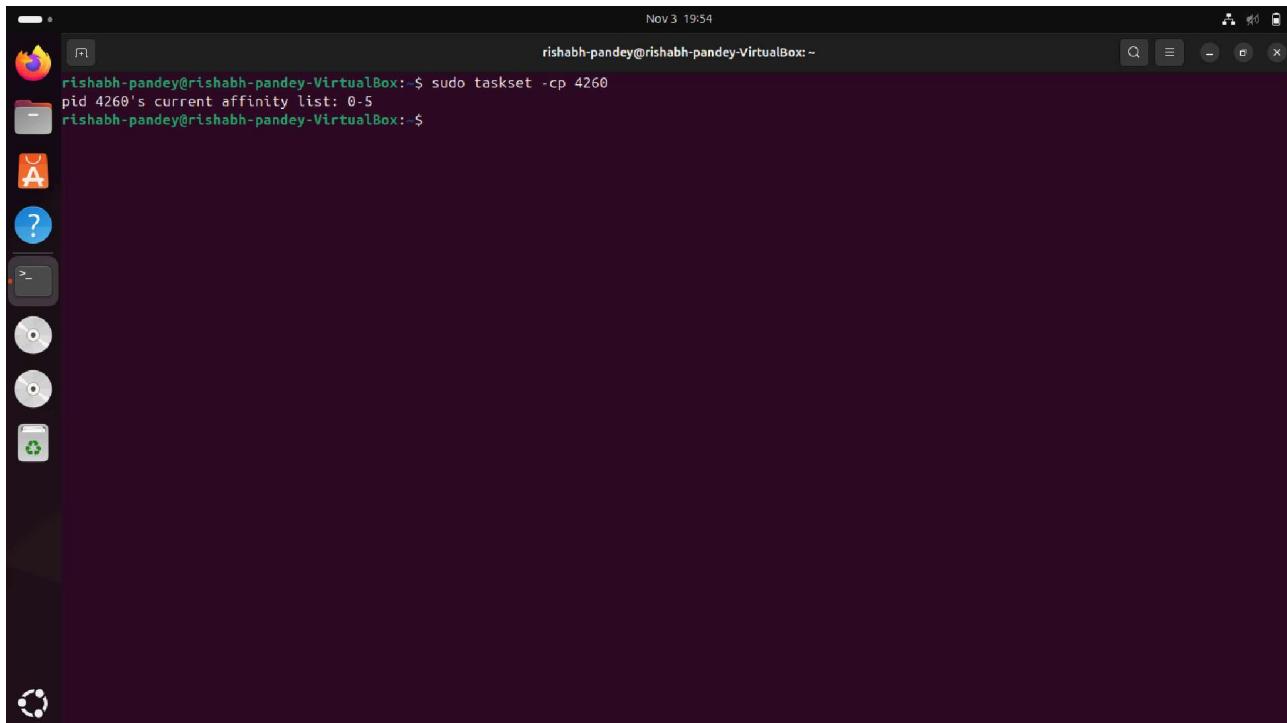
🔧 5. CPU Affinity — Binding a Process to Specific Cores

🔍 Inspect Current CPU Affinity

```
taskset -cp 3050
```



📷 Output



```
Nov 3 19:54
rishabh-pandey@rishabh-pandey-VirtualBox: $ sudo taskset -cp 4260
pid 4260's current affinity list: 0-5
rishabh-pandey@rishabh-pandey-VirtualBox: $
```

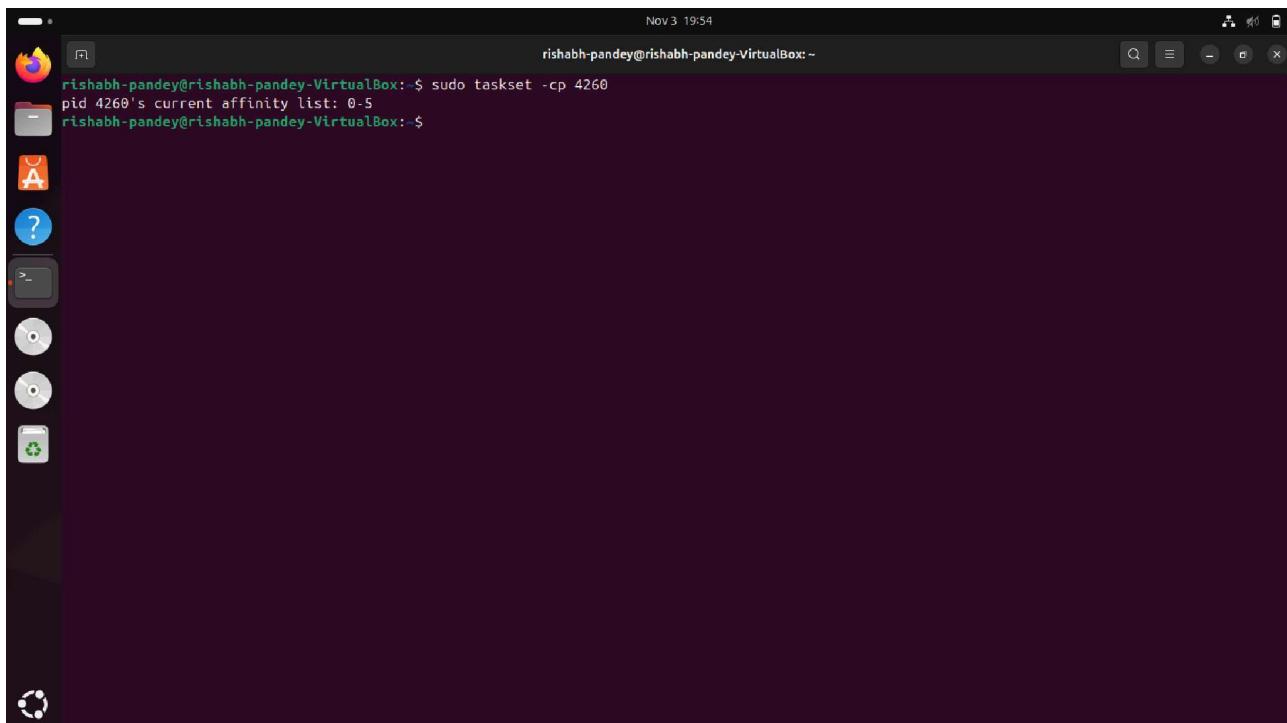
🎯 Bind the Process Strictly to Core 1

```
taskset -cp 1 3050
```



👉 Useful for performance isolation, cache locality experiments, or preventing contention.

📷 Output



```
Nov 3 19:54
rishabh-pandey@rishabh-pandey-VirtualBox: $ sudo taskset -cp 4260
pid 4260's current affinity list: 0-5
rishabh-pandey@rishabh-pandey-VirtualBox: $
```

📁 6. I/O Scheduling Priority

🔧 Command

```
ionice -c 3 -p 3050
```



👉 Class 3 (idle) means the process performs disk I/O *only when the system is otherwise idle* — excellent for background jobs.

📷 Output

The screenshot shows a terminal window on a dark-themed desktop environment. The terminal title bar says "Nov 3 19:59". The command "rishabh-pandey@rishabh-pandey-VirtualBox: ~" is visible. The user runs the command "sudo ionice -c 3 -p 4295". The terminal window has a standard Linux interface with icons for file operations like copy, paste, and cut.

```
rishabh-pandey@rishabh-pandey-VirtualBox: ~$ sudo ionice -c 3 -p 4295
```

📝 7. Inspecting File Descriptors of a Process

🔧 Command

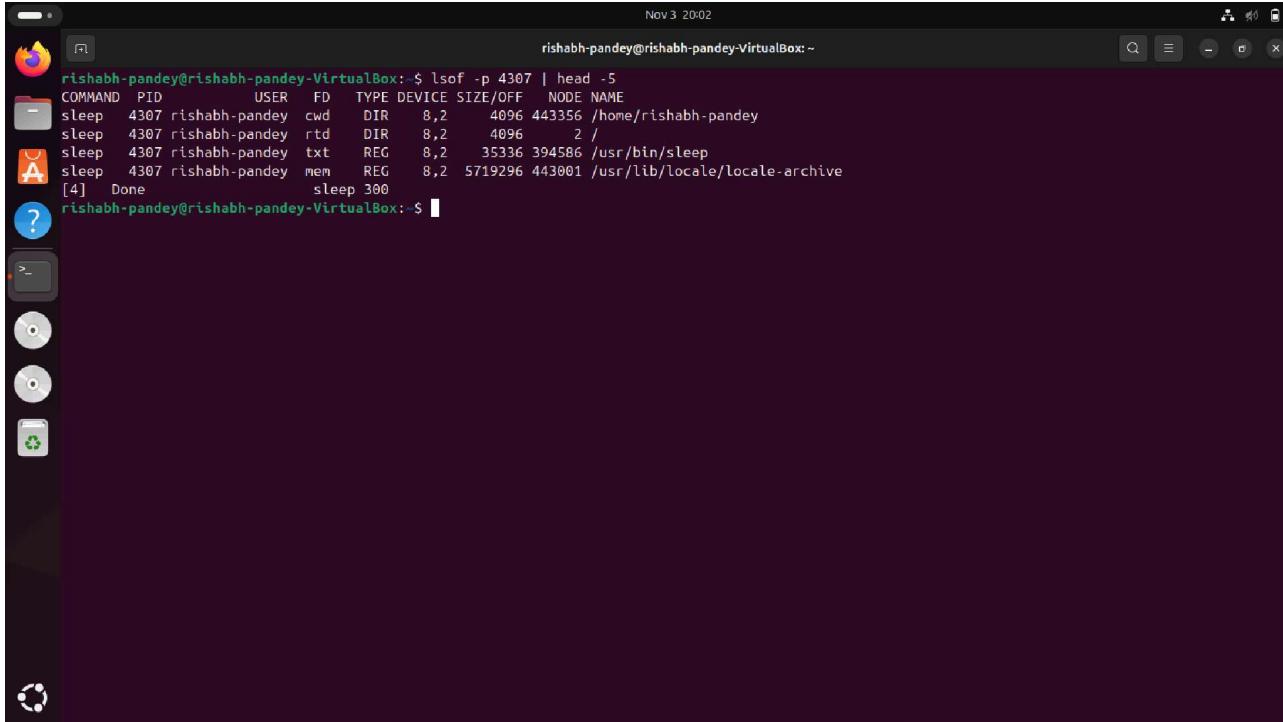
```
lsof -p 3050 | head -5
```



✳️ Conceptual Insight

A process is fundamentally defined by the **resources it holds**. `lsof` reveals open files, sockets, pipes, and more — invaluable for debugging stuck I/O or memory-leaking daemons.

📸 Output



The screenshot shows a terminal window titled "Nov 3 20:02" with the command `rishabh-pandey@rishabh-pandey-VirtualBox: $ lsof -p 4307 | head -5` running. The output lists five processes associated with PID 4307:

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE/OFF	NODE NAME
sleep	4307	rishabh-pandey	cwd	DIR	8,2	4096	443356 /home/rishabh-pandey
sleep	4307	rishabh-pandey	rtd	DIR	8,2	4096	2 /
sleep	4307	rishabh-pandey	txt	REG	8,2	35336	394586 /usr/bin/sleep
sleep	4307	rishabh-pandey	mem	REG	8,2	5719296	443001 /usr/lib/locale/locale-archive

[4] Done sleep 300

rishabh-pandey@rishabh-pandey-VirtualBox: \$

🐛 8. Tracing System Calls

🔧 Command

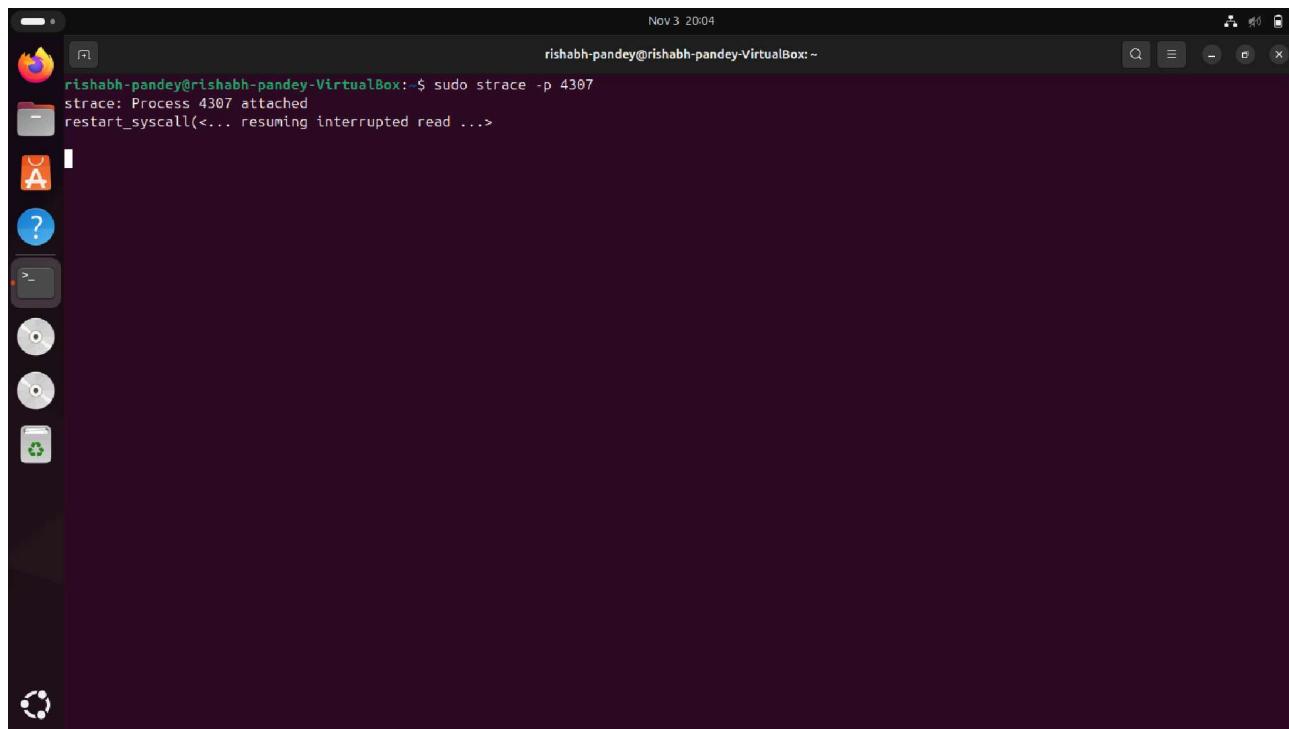
```
strace -p 3050
```



🛠️ Conceptual Insight

This attaches a live syscall-level microscope to the process. Perfect for reverse-engineering behavior, debugging failures, or understanding program–kernel interactions.

📸 Output



A screenshot of a Linux desktop environment showing a terminal window. The terminal window has a dark background and contains the following text:

```
rishabh-pandey@rishabh-pandey-VirtualBox: ~$ sudo strace -p 4307
strace: Process 4307 attached
restart_syscall(<... resuming interrupted read ...>
```



9. Identifying Which Process Owns a Network Port



Command

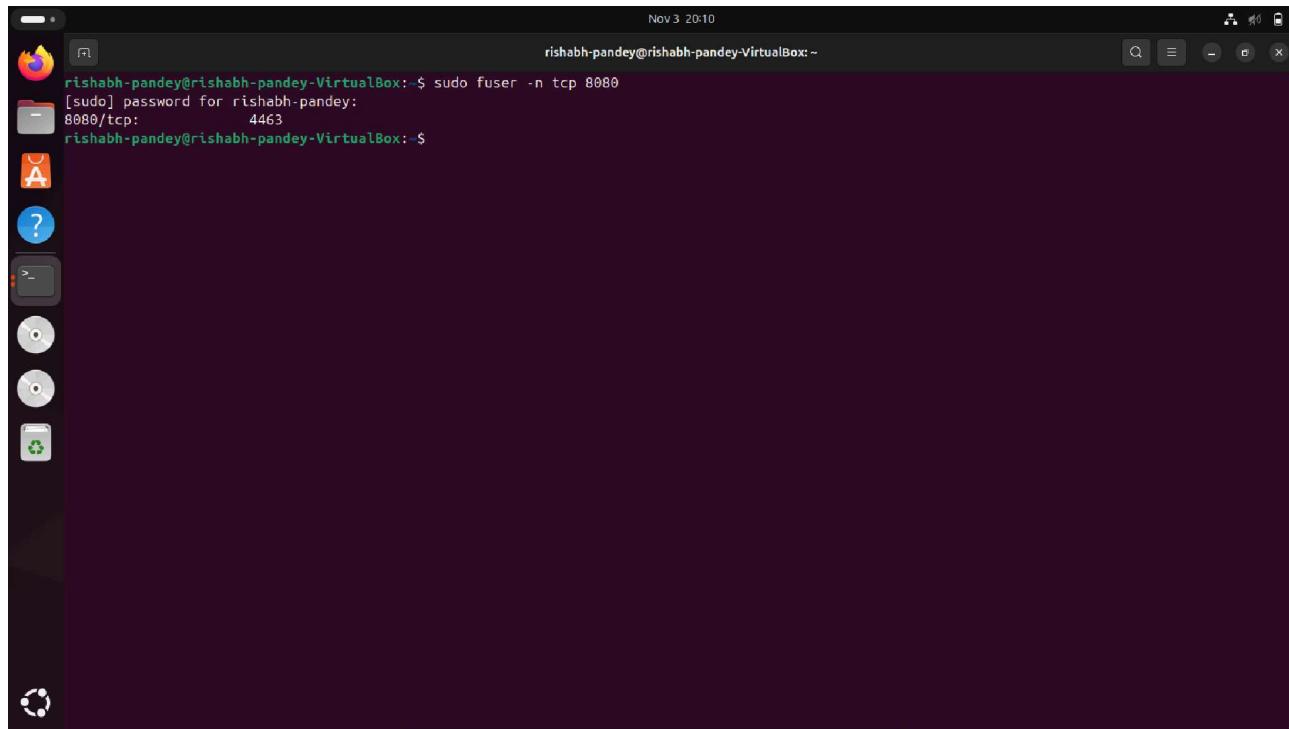
```
sudo fuser -n tcp 8080
```



👉 If PID 4321 appears, that's your port consumer — often helpful for debugging services, conflicts, or ghost processes.



Output



A screenshot of a Linux desktop environment showing a terminal window. The terminal window has a dark background and contains the following text:

```
rishabh-pandey@rishabh-pandey-VirtualBox: $ sudo fuser -n tcp 8080
[sudo] password for rishabh-pandey:
8080/tcp:        4463
rishabh-pandey@rishabh-pandey-VirtualBox: $
```

The terminal window is titled "Terminal" and shows the date and time as "Nov 3 20:10". The desktop interface includes a vertical dock on the left with icons for a browser, file manager, terminal, and system settings.



10. Per-Process Performance Statistics

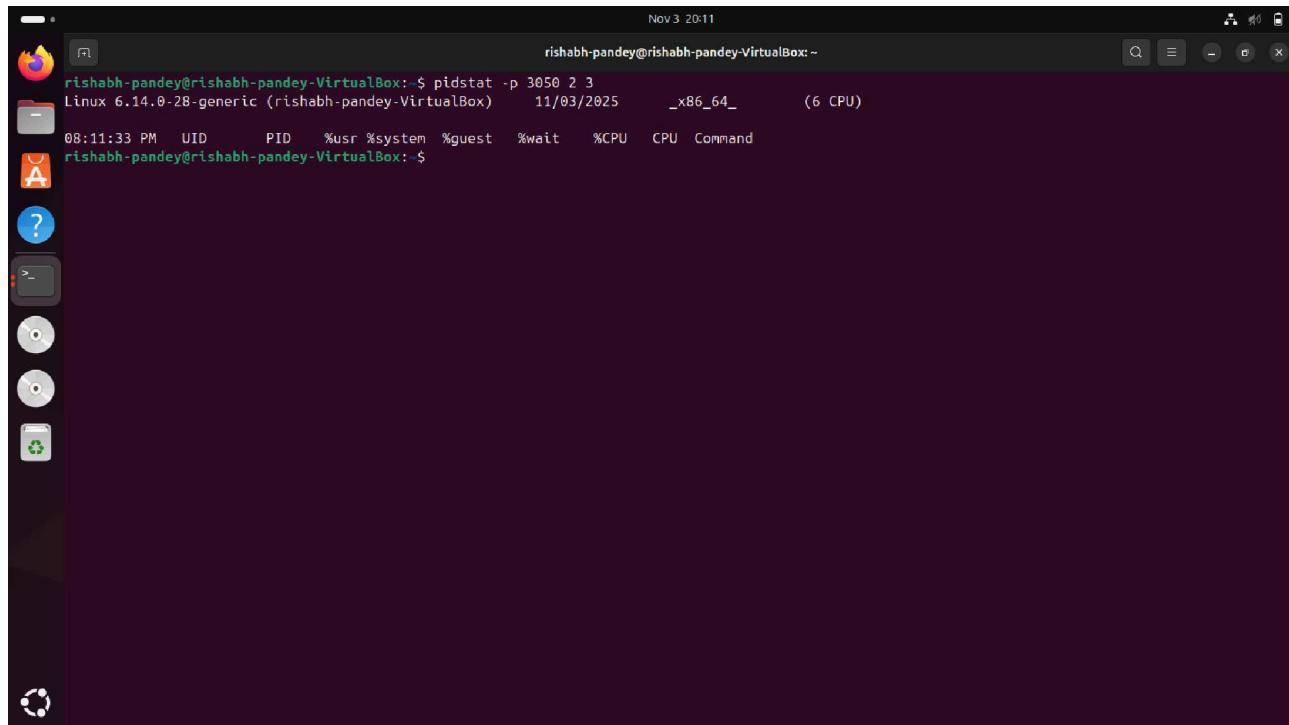


Command

```
pidstat -p 3050 2 3
```



👉 Samples CPU/IO usage every **2 seconds**, repeating **3 times** — a lightweight profiler without full tracing overhead.



A screenshot of a Linux desktop environment. On the left is a dark-themed file manager sidebar with icons for Home, Applications, Help, and a trash can. The main window is a terminal window titled 'rishabh-pandey@rishabh-pandey-VirtualBox'. It shows the command 'pidstat -p 3050 2 3' being run, displaying CPU usage statistics for PID 3050 over three iterations. The terminal window has a dark background and light-colored text. The title bar includes the date and time: 'Nov 3 20:11'.

📸 Output

🔒 11. Resource Governance with Control Groups (cgroups)

🛠️ Create a Dedicated cgroup

```
sudo cgcreate -g cpu,memory:/testgroup
```



🚦 Apply Resource Limits

```
echo 50000 | sudo tee /sys/fs/cgroup/cpu/testgroup/cpu.cfs_quota_us  
echo 100M | sudo tee /sys/fs/cgroup/memory/testgroup/memory.limit_in_bytes
```



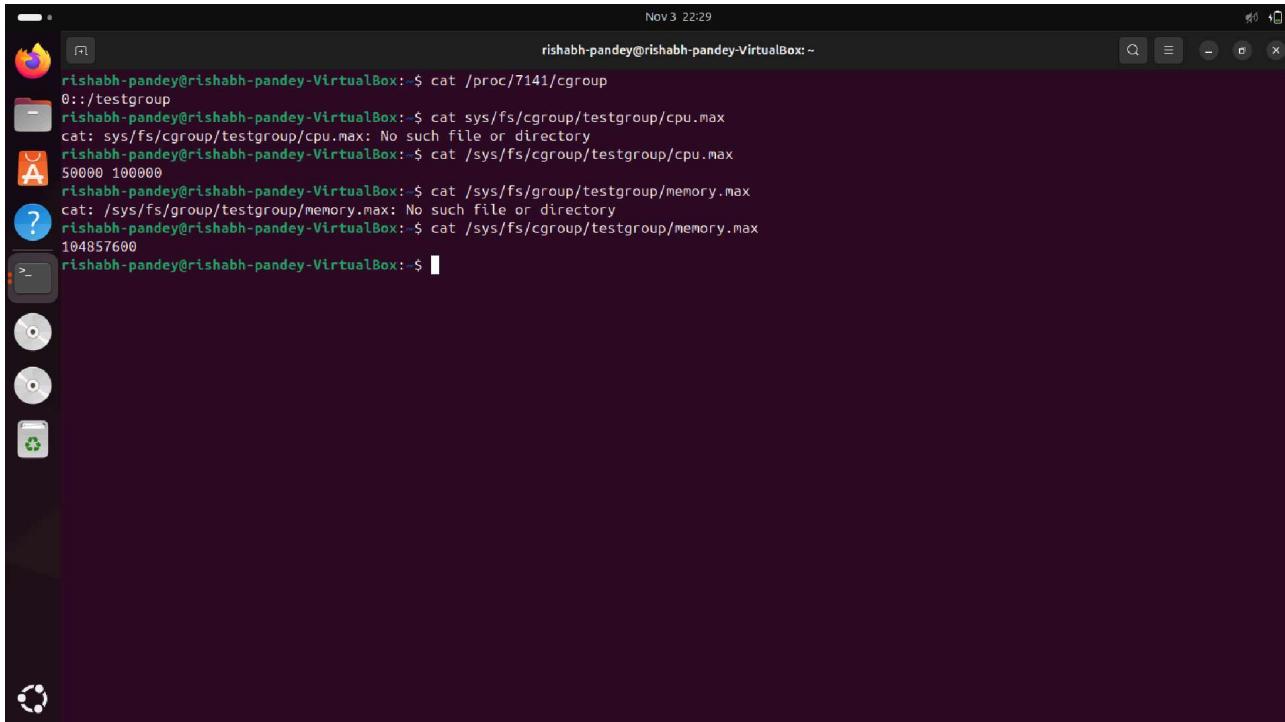
➕ Add PID 3050 to the Group

```
echo 3050 | sudo tee /sys/fs/cgroup/cpu/testgroup/cgroup.procs
```



👉 cgroups offer fine-grained, kernel-enforced governance — far beyond what nice can control.

Output



A screenshot of a terminal window titled "Output". The terminal shows a session on a VirtualBox machine named "rishabh-pandey@rishabh-pandey-VirtualBox". The user runs several commands to check cgroup settings:

```
Nov 3 22:29
rishabh-pandey@rishabh-pandey-VirtualBox: $ cat /proc/7141/cgroup
0:::/testgroup
rishabh-pandey@rishabh-pandey-VirtualBox: $ cat sys/fs/cgroup/testgroup/cpu.max
cat: sys/fs/cgroup/testgroup/cpu.max: No such file or directory
rishabh-pandey@rishabh-pandey-VirtualBox: $ cat /sys/fs/cgroup/testgroup/cpu.max
50000 10000
rishabh-pandey@rishabh-pandey-VirtualBox: $ cat /sys/fs/group/testgroup/memory.max
cat: /sys/fs/group/testgroup/memory.max: No such file or directory
rishabh-pandey@rishabh-pandey-VirtualBox: $ cat /sys/fs/cgroup/testgroup/memory.max
104857600
rishabh-pandey@rishabh-pandey-VirtualBox: $
```

🎯 12. Alternatives & Complements to `nice` / `renice`

1 Real-Time Scheduling — `chrt`

```
sudo chrt -f 50 sleep 1000
chrt -p <pid>
```



2 Disk I/O Prioritization — `ionice`

```
ionice -c 2 -n 7 tar -czf backup.tar.gz /home
```



3 CPU Affinity — `taskset`

```
taskset -c 1 firefox
```



4 cgroups for Granular Constraints

```
sudo cgcreate -g cpu,memory:/lowprio
echo 20000 | sudo tee /sys/fs/cgroup/cpu/lowprio/cpu.cfs_quota_us
echo 200M   | sudo tee /sys/fs/cgroup/memory/lowprio/memory.limit_in_bytes
echo 1234 | sudo tee /sys/fs/cgroup/cpu/lowprio/cgroup.procs
```



5 systemd-integrated Resource Control — `systemd-run`

```
systemd-run --scope -p CPUWeight=200 stress --cpu 4
```



6 Scheduling with `schedtool`

```
sudo schedtool -R -p 10 <pid>
```



🔗 Consolidated Summary

Tool	Focus Area	Relation to <code>nice</code>
<code>chrt</code>	Real-time scheduling policies	Alternative (stronger)
<code>ionice</code>	Disk I/O prioritization	Complementary
<code>taskset</code>	CPU affinity	Complementary
<code>cgroups</code>	Kernel-level resource governance	Superset of capabilities
<code>systemd-run</code>	systemd + cgroup orchestration	Alternative + orchestration
<code>schedtool</code>	Custom scheduling classes	Alternative