



rish4name / Lms_Linux



<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

In

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-slub_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 ?        I    18:40   0:00 [kworker/2:0-rcu_gp]

```

2. Visualizing the Process Hierarchy

Command

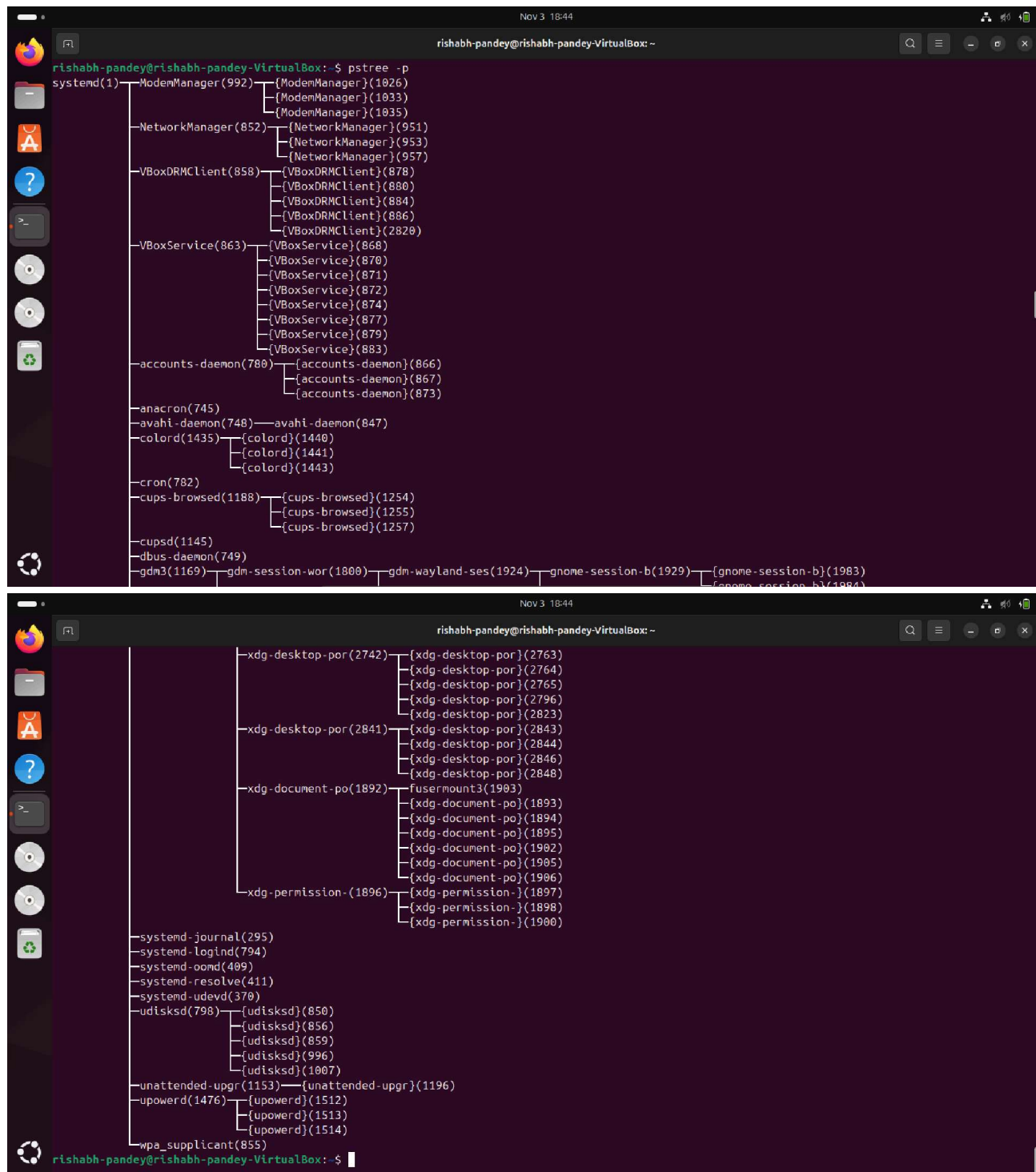
`ps tree -p`



Conceptual Insight

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

Output



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

```

Nov 3 19:12
rishabh-pandey@rishabh-pandey-VirtualBox: ~
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  S  %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-netns
   11 root      0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-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

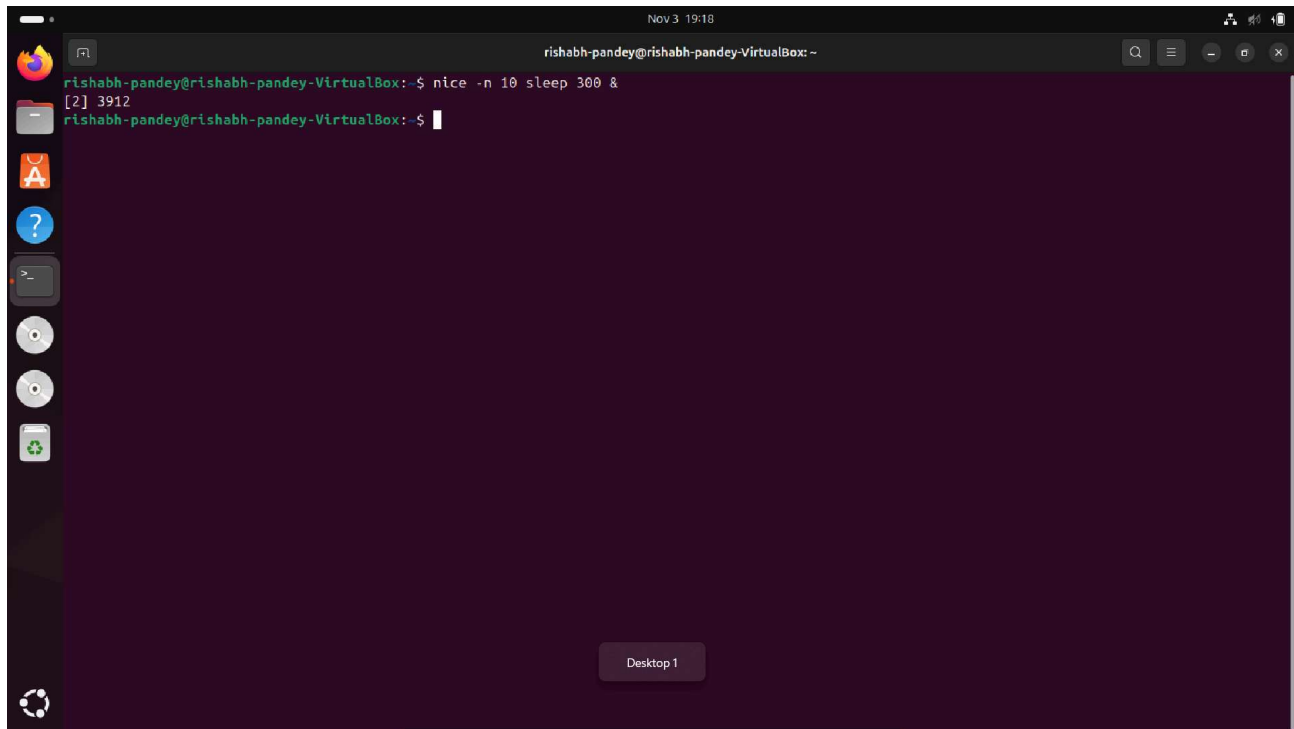
🟢 Start a Process at Lower Priority

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



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

📷 Output



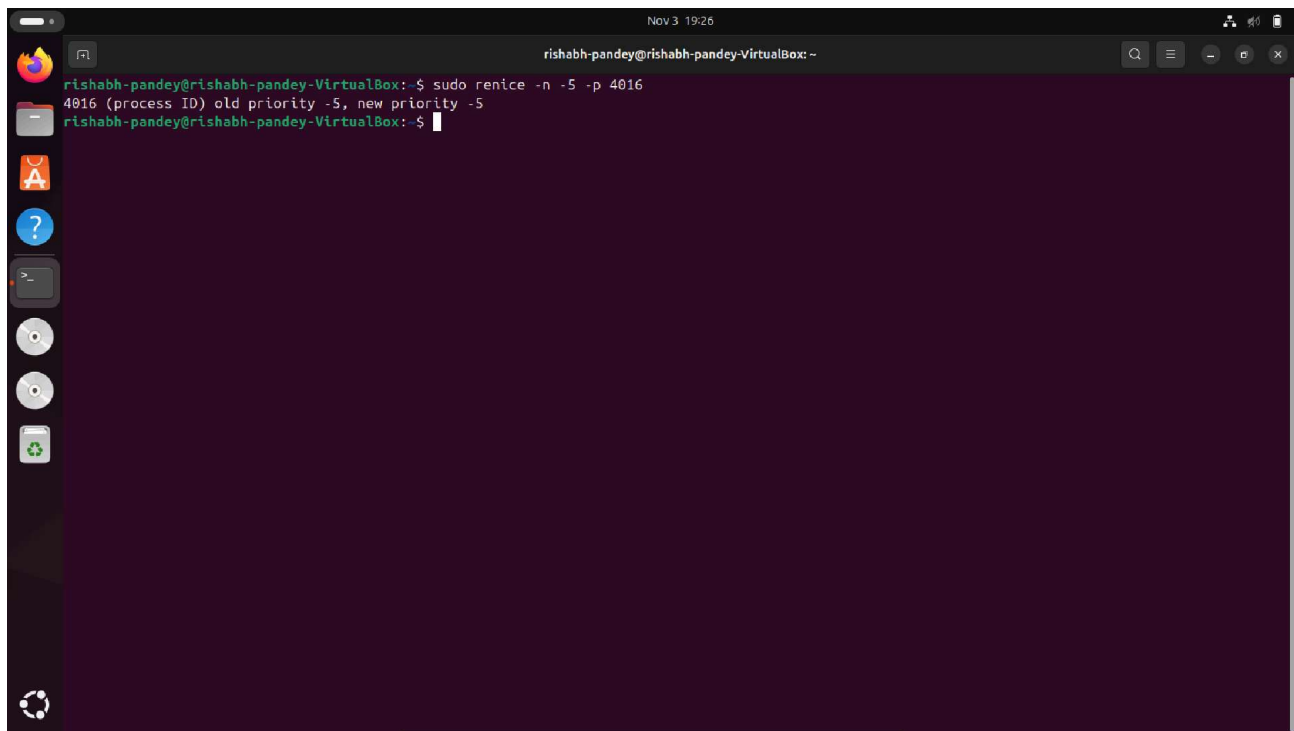
Reprioritize an Existing Process

```
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



```
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: ~$
```

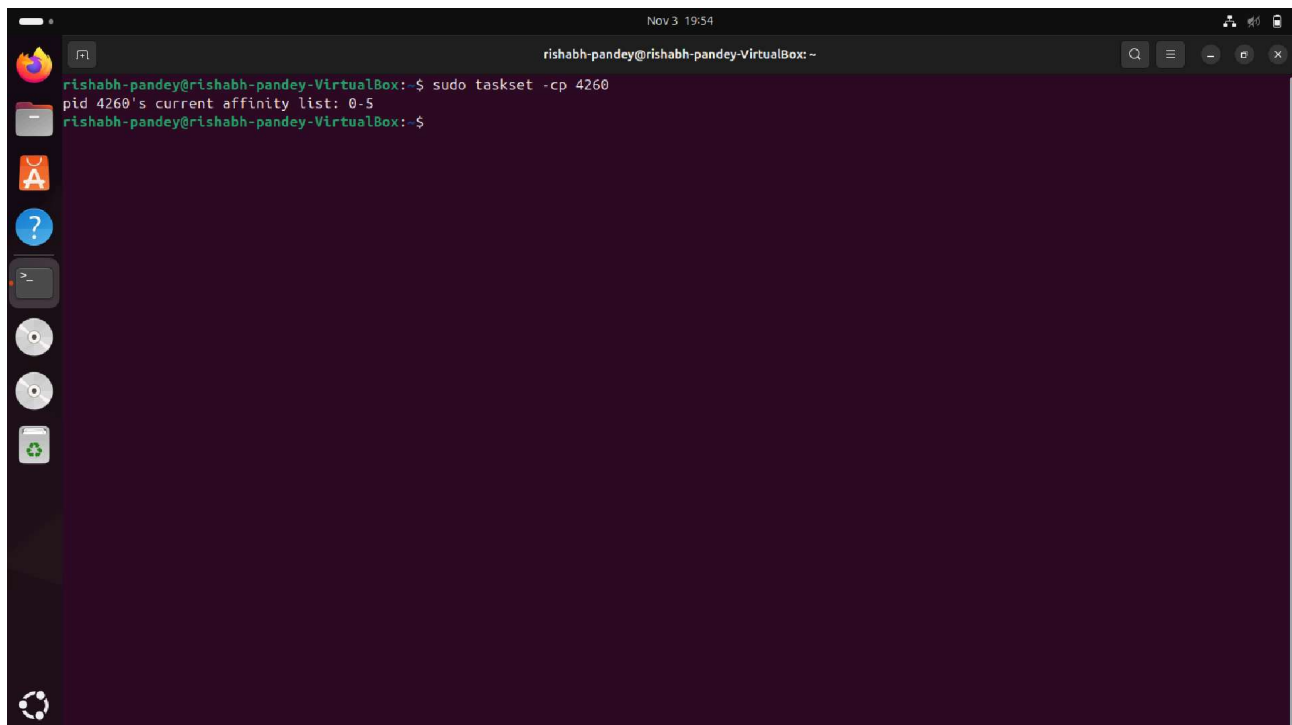
5. CPU Affinity — Binding a Process to Specific Cores

Inspect Current CPU Affinity

```
taskset -cp 3050
```



Output

A terminal window titled 'rishabh-pandey@rishabh-pandey-VirtualBox: ~' with a dark purple background. The terminal shows the command 'sudo taskset -cp 4260' being executed. The output is 'pid 4260's current affinity list: 0-5'. The prompt returns to 'rishabh-pandey@rishabh-pandey-VirtualBox: ~\$'. On the left side of the terminal, there is a vertical dock with icons for a file manager, application store, help, terminal, and other system utilities.

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

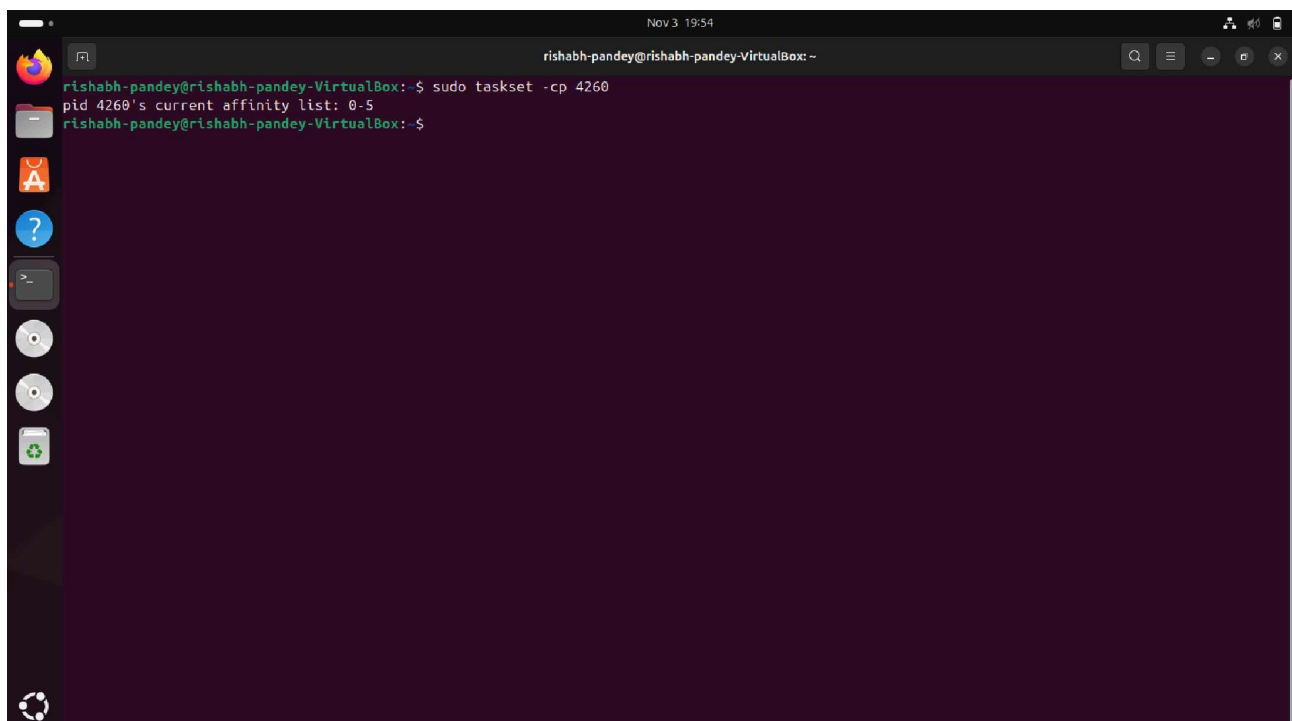
Bind the Process Strictly to Core 1

```
taskset -cp 1 3050
```



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

Output

A terminal window titled 'rishabh-pandey@rishabh-pandey-VirtualBox: ~' with a dark purple background. The terminal shows the command 'sudo taskset -cp 4260' being executed. The output is 'pid 4260's current affinity list: 0-5'. The prompt returns to 'rishabh-pandey@rishabh-pandey-VirtualBox: ~\$'. On the left side of the terminal, there is a vertical dock with icons for a file manager, application store, help, terminal, and other system utilities.


```
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

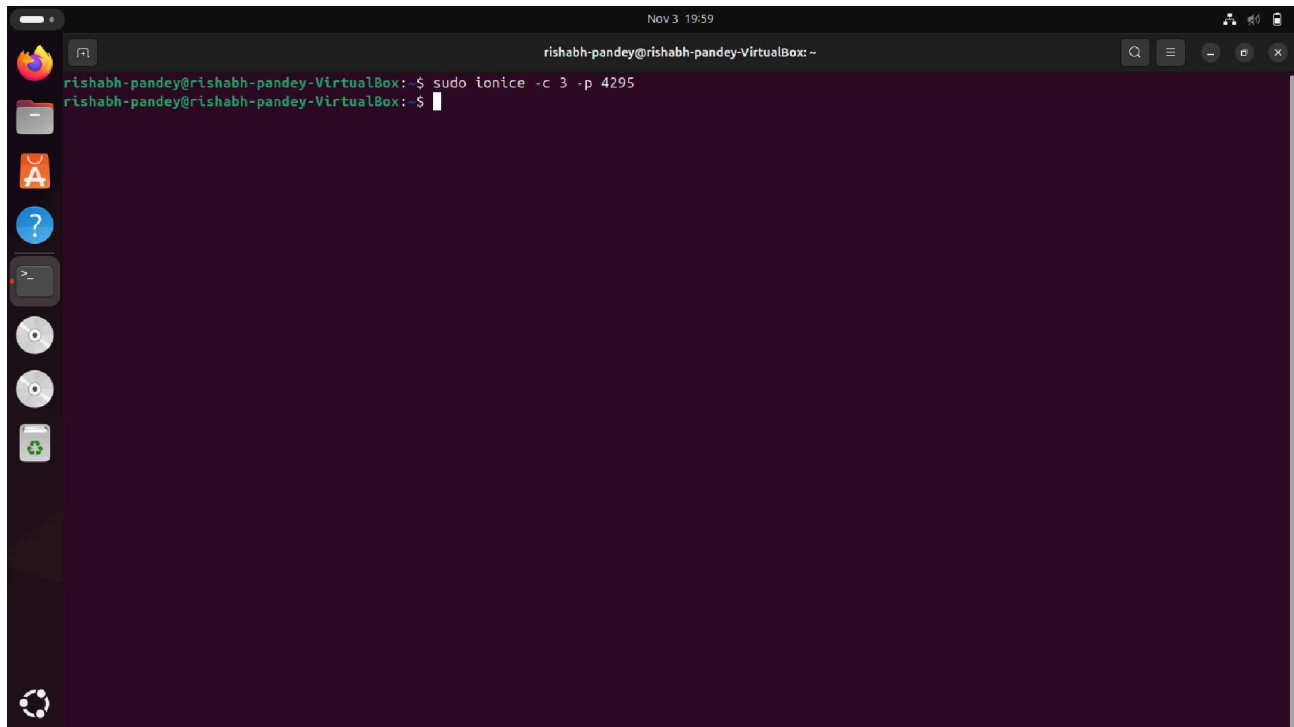
Command

```
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



```
Nov 3 19:59
rishabh-pandey@rishabh-pandey-VirtualBox: ~
rishabh-pandey@rishabh-pandey-VirtualBox: ~$ sudo ionice -c 3 -p 4295
rishabh-pandey@rishabh-pandey-VirtualBox: ~$
```

7. Inspecting File Descriptors of a Process

Command

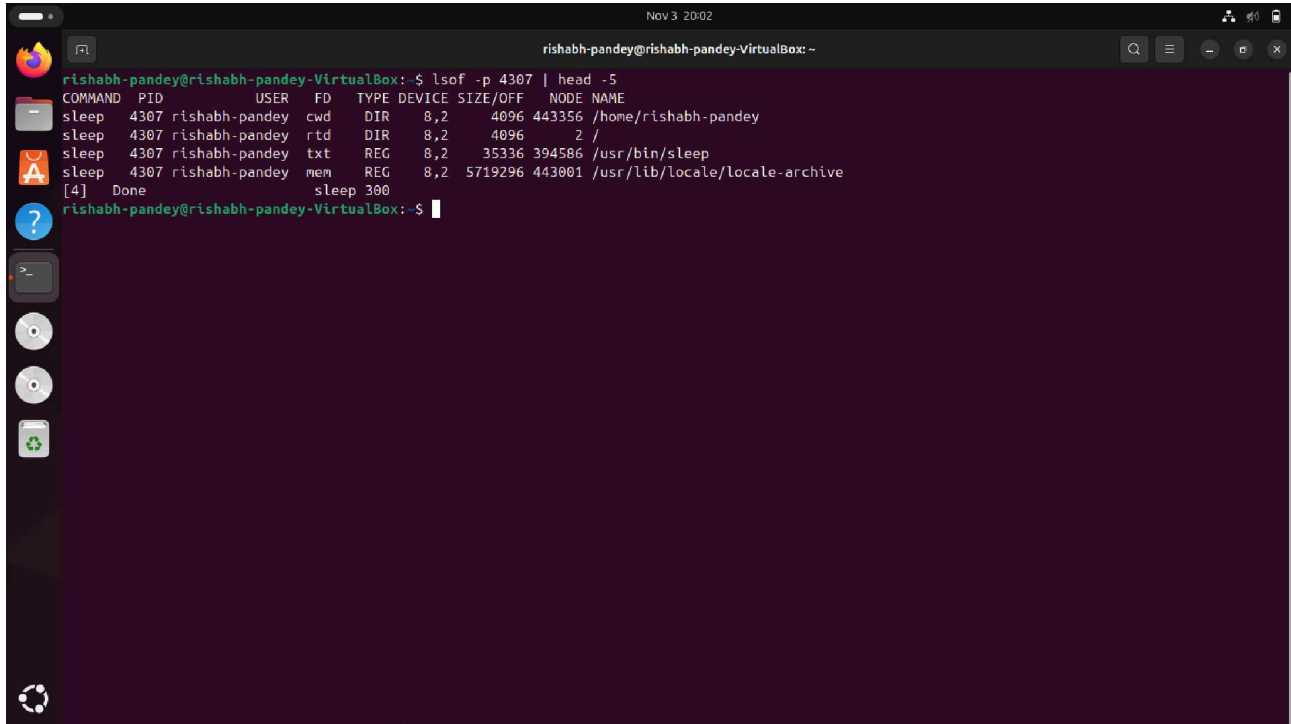
```
lsof -p 3050 | head -5
```



Conceptual Insight

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

Output



```
Nov 3 20:02
rishabh-pandey@rishabh-pandey-VirtualBox: ~
rishabh-pandey@rishabh-pandey-VirtualBox:~$ lssof -p 4307 | head -5
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 443801 /usr/lib/locale/locale-archive
[4] Done
sleep 300
rishabh-pandey@rishabh-pandey-VirtualBox:~$
```

8. Tracing System Calls

Command

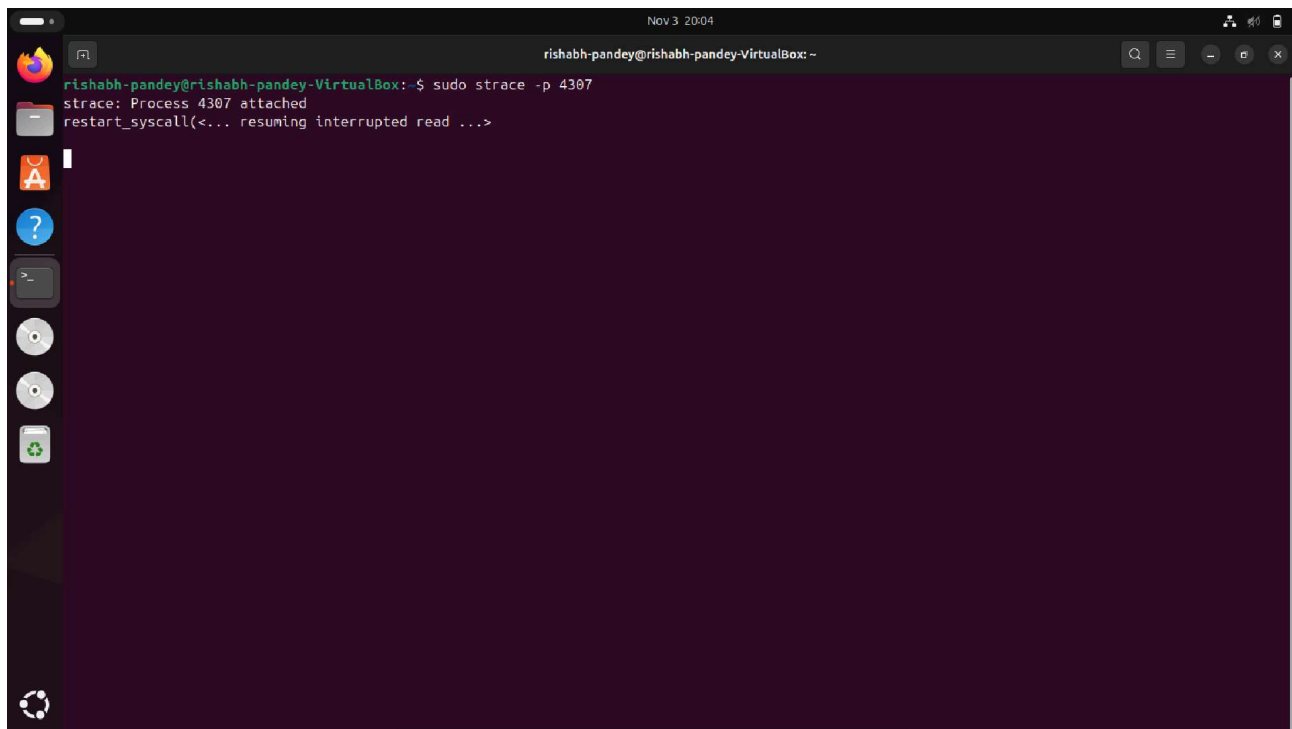
```
strace -p 3050
```



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



```
Nov 3 20:04
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

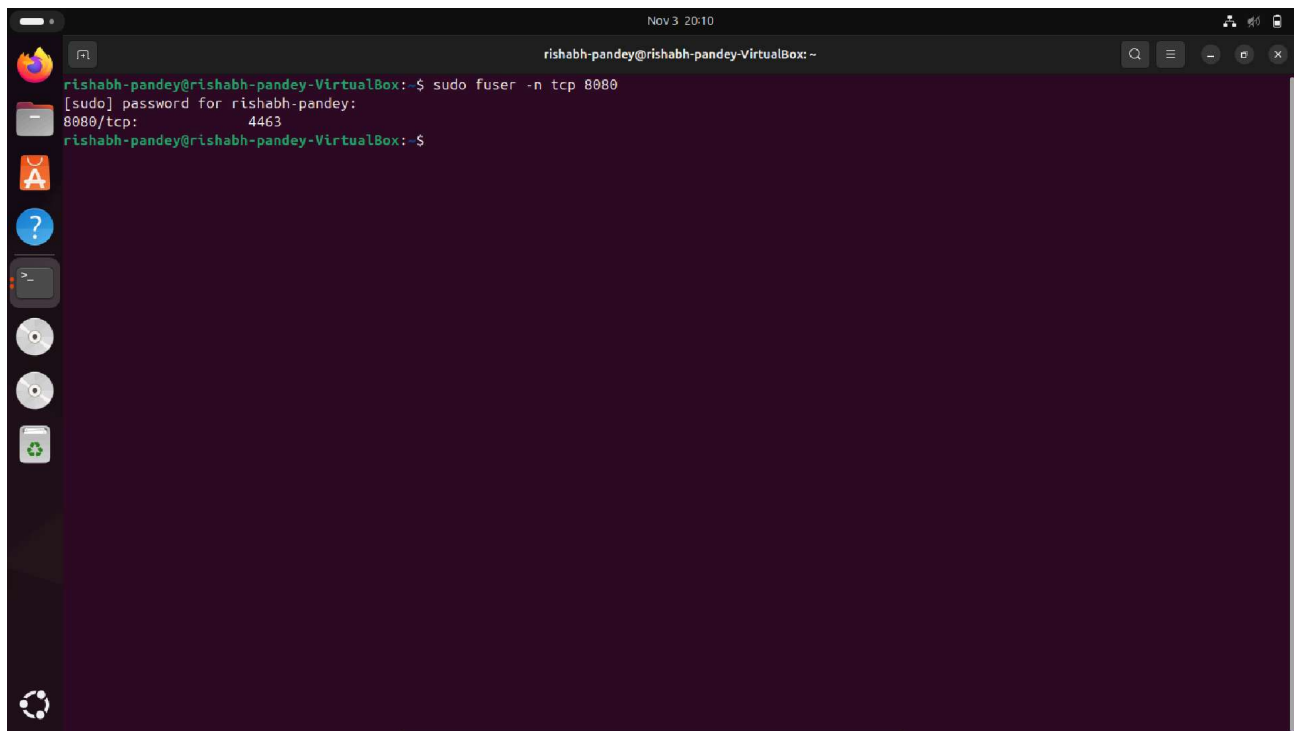
```
sudo fuser -n tcp 8080
```



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



Output



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



10. Per-Process Performance Statistics

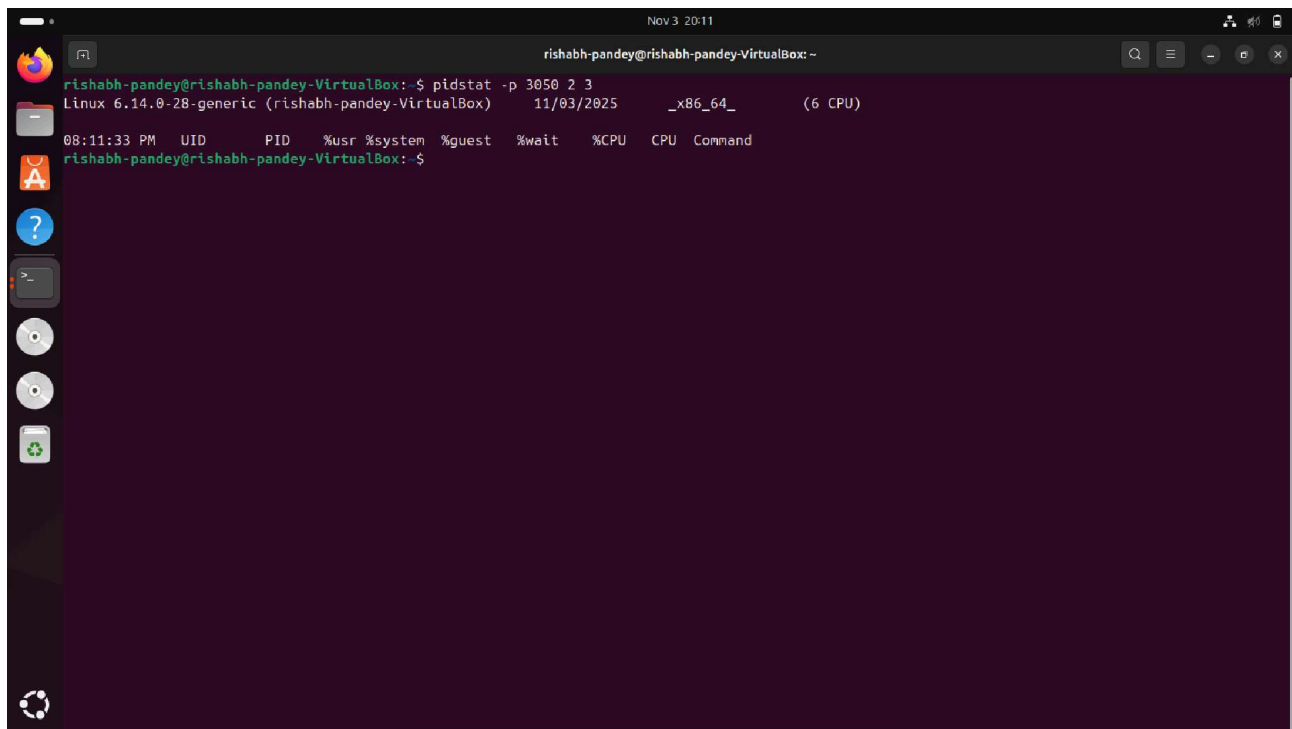


Command

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



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



```
rishabdh-pandey@rishabdh-pandey-VirtualBox: ~  
$ pidstat -p 3050 2 3  
Linux 6.14.0-28-generic (rishabdh-pandey-VirtualBox) 11/03/2025 _x86_64_ (6 CPU)  
08:11:33 PM UID      PID    %usr %system %guest  %wait   %CPU   CPU  Command  
rishabdh-pandey@rishabdh-pandey-VirtualBox: $
```

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
```



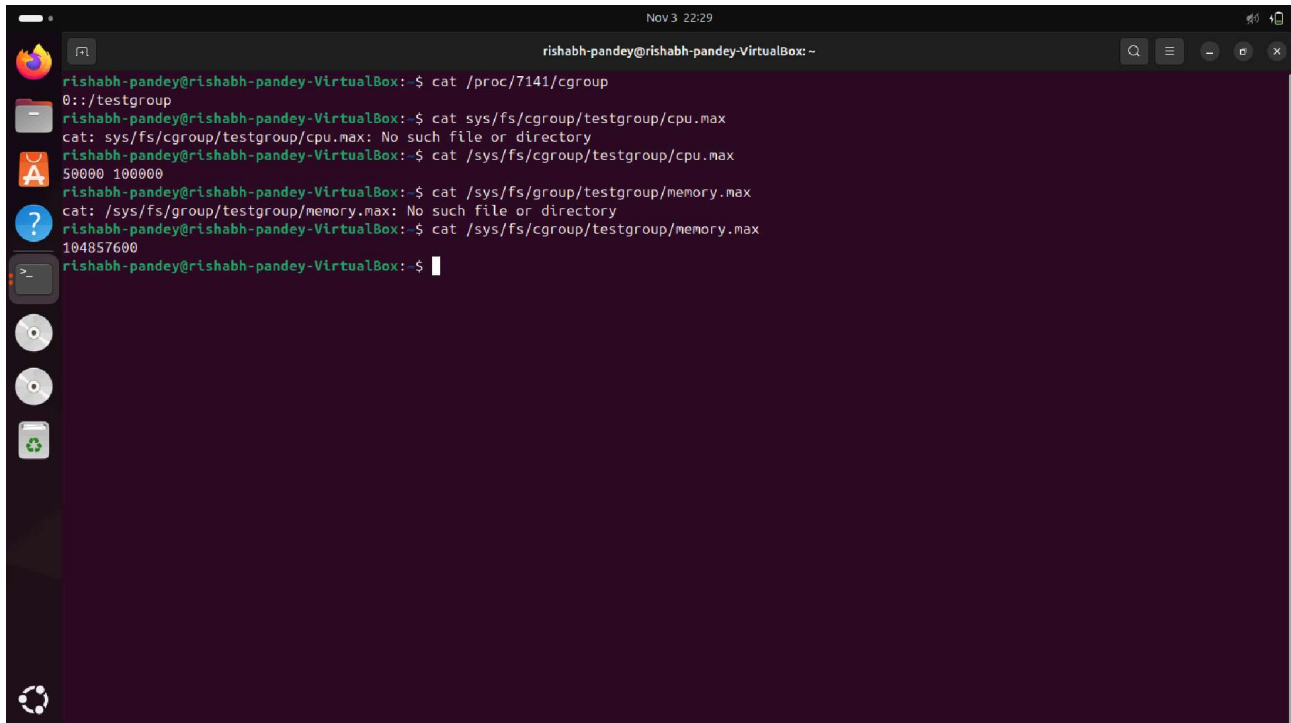
Add PID 3050 to the Group

```
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



```
Nov 3 22:29
rishabh-pandey@rishabh-pandey-VirtualBox: ~
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
500000 100000
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