Linux_Process_Management_Guide

Linux Process Management Guide

Author: Divyanshi Thapa

Date: September 26, 2025

---

ps accepts options in various formats:

- not preceded with a dash (BSD style)
- preceded with a dash (UNIX style)
- preceded with two dashes (GNU style)

Table of Contents

1. Viewing All Processes
2. Simple filtering
3. Listing threads
4. Listing Child processes
5. Controlling the Output
6. Process Tree
7. Real-Time Process Monitoring
8. Adjusting Process Priority
9. CPU Affinity (Bind Process to CPU Core)
10. I/O Scheduling Priority
11. File Descriptors Used by a Process
12. Trace System Calls of a Process
13. Find Process Using a Port
14. Per-Process Statistics
15. Control Groups for Resource Limits
16. Alternatives to nice / renice
17. Cheat Sheet

NOTE: To find a process by its name like "sleep 3000": `sudo pgrep -f "sleep 1000"`
NOTE: To kill a process if you know its name like "sleep 300" : `sudo pkill -f "sleep 300"`

1. List Processes

Command:

```
ps aux
```

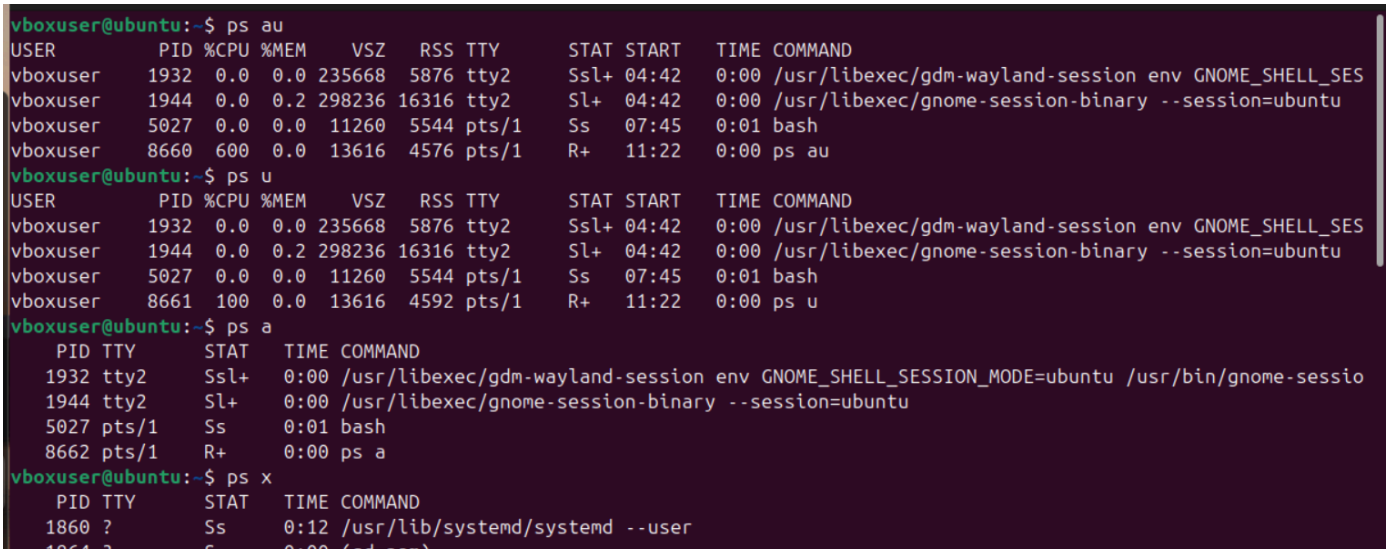
Explanation:

- **ps** → process status
- **a** → show processes for all users
- **u** → show user/owner of process
- **x** → includes off terminal commands

Example Output:

```
USER         PID  %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.1 167500  1100 ?        Ss   Sep25   0:05 /sbin/init
vibhu       1234  1.2  1.5 274532 15632 ?        Sl   10:15   0:12 /usr/bin/python3
script.py
mysql       2001  0.5  2.0 450000 20988 ?        Ssl  Sep25   1:02 /usr/sbin/mysqld
```

Sample code:



Command : `ps -e`

Note: see a more detailed output by using the -f in last.
Note : -f adds on the additional info in all the other commands.

2. Simple filtering

Command: `ps -C processname`

searching for a particular process by name with the -C option.

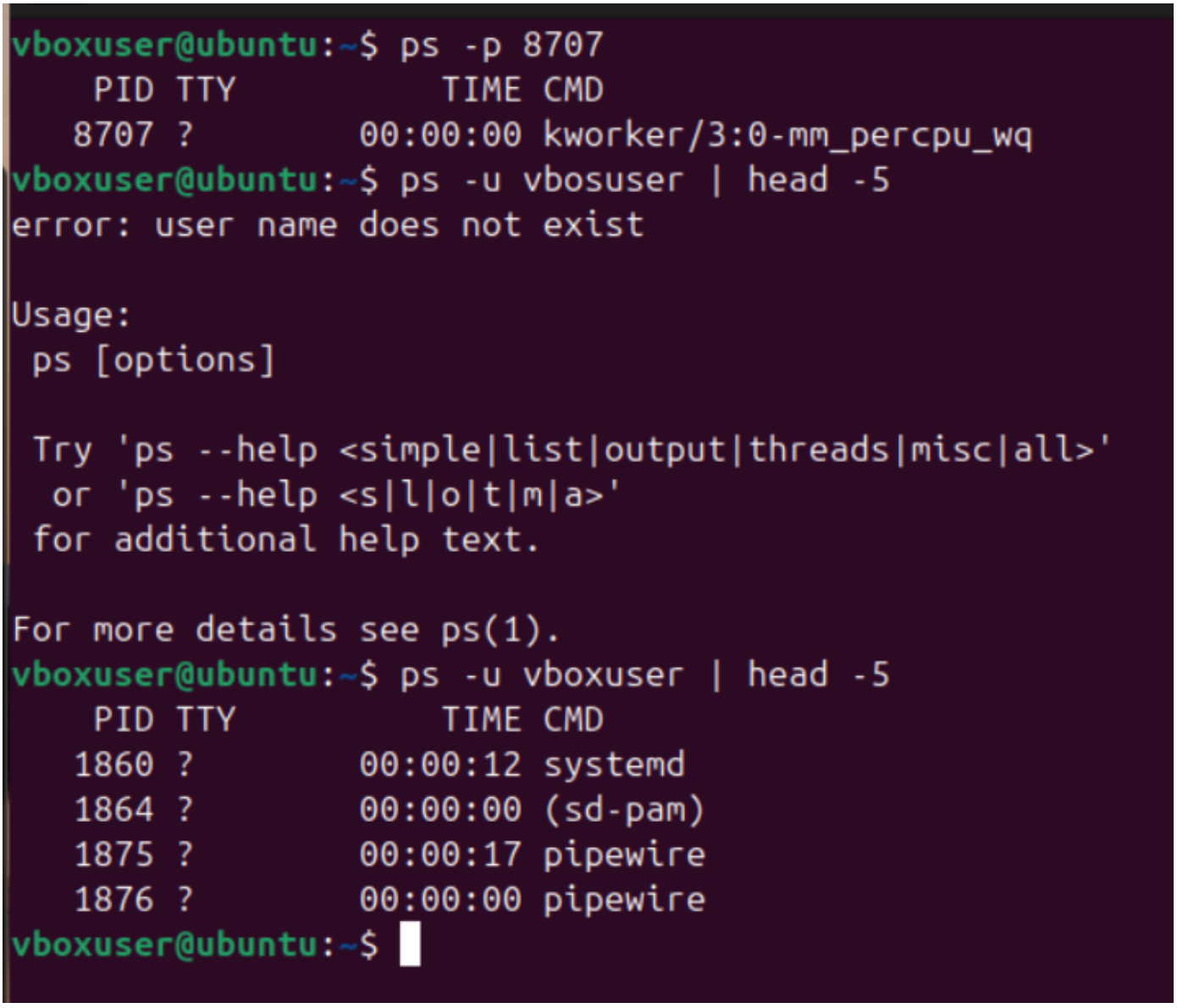
Command: `ps -p processid`

filter based on a list of process ids using the -p flag.

Command:; `ps -u username`

search by the user name by specifying it in the -u option.

Sample code:



3. Listing Threads

Command: `ps -C gedit -L`

- Todo: search a process named gedit and list all its threads.
  Note : you can ask for a more informative output then :
  `ps -C gedit -L -f`
- total number of threads of the process in the NLWP column.

Sample code:

4. Listing Child Processes

see the spawned child processes rather than the threads.

Command: `ps -e -H`

- H - for heirarchy
- we can filter by session id (SID) only and not by pid or process name.
  For that: `ps -g sid -H`

or

`ps -s sid -H`

Sample code:

```
vboxuser@ubuntu:~$ ps -e -H | head -5
    PID TTY          TIME CMD
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00   pool_workqueue_release
      4 ?        00:00:00   kworker/R-rcu_gp
      5 ?        00:00:00   kworker/R-sync_wq
vboxuser@ubuntu:~$ ps -e -H -o pid,cmd | head -5
    PID CMD
      2 [kthreadd]
      3   [pool_workqueue_release]
      4   [kworker/R-rcu_gp]
      5   [kworker/R-sync_wq]
vboxuser@ubuntu:~$ ps -C bash -o sid
    SID
   2706
vboxuser@ubuntu:~$ ps -g 2706 -H | head -5
    PID TTY          TIME CMD
   2706 pts/0    00:00:00 bash
   2839 pts/0    00:00:00   ps
   2840 pts/0    00:00:00   head
vboxuser@ubuntu:~$
```

How to see my session id?

`ps -C gedit -o sid`

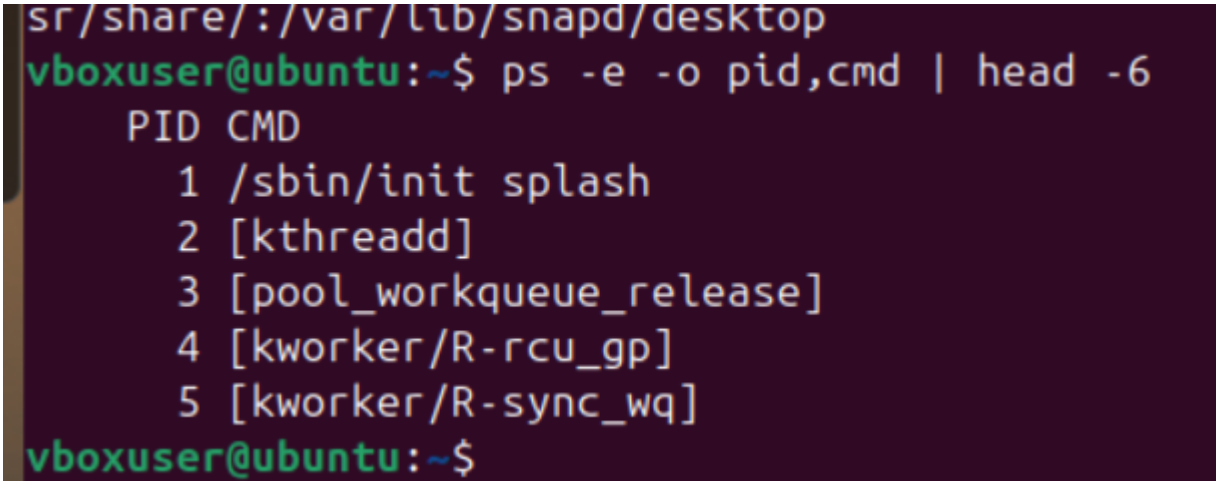
Note:

session id is equal to the process id that started the session — also called the session leader.

5. Controlling the Output

control which columns are printed with the help of the -o flag (order oriented).
`ps -C gedit -o pid,tty`

Sample code:



6. Process Tree

Command:

```
pstree -p
```

Explanation:

Shows parent-child process relationships.

- [] denote identical branches.
- { } denote child thread.

Example Output:

```
systemd(1)─┬─NetworkManager(778)
           ├─sshd(895)─┬─sshd(1023)───bash(1024)───pstree(1101)
           ├─mysqld(2001)
           └─python3(1234)
```

Sample code:

```
            └─wpa_supplicant(863)
vboxuser@ubuntu:~$ pstree -p | head -5
systemd(1)-+-ModemManager(936)-+-{ModemManager}(965)
           |                   |-{ModemManager}(974)
           |                   `-{ModemManager}(977)
           |-NetworkManager(858)-+-{NetworkManager}(935)
           |                     |-{NetworkManager}(938)
vboxuser@ubuntu:~$
```

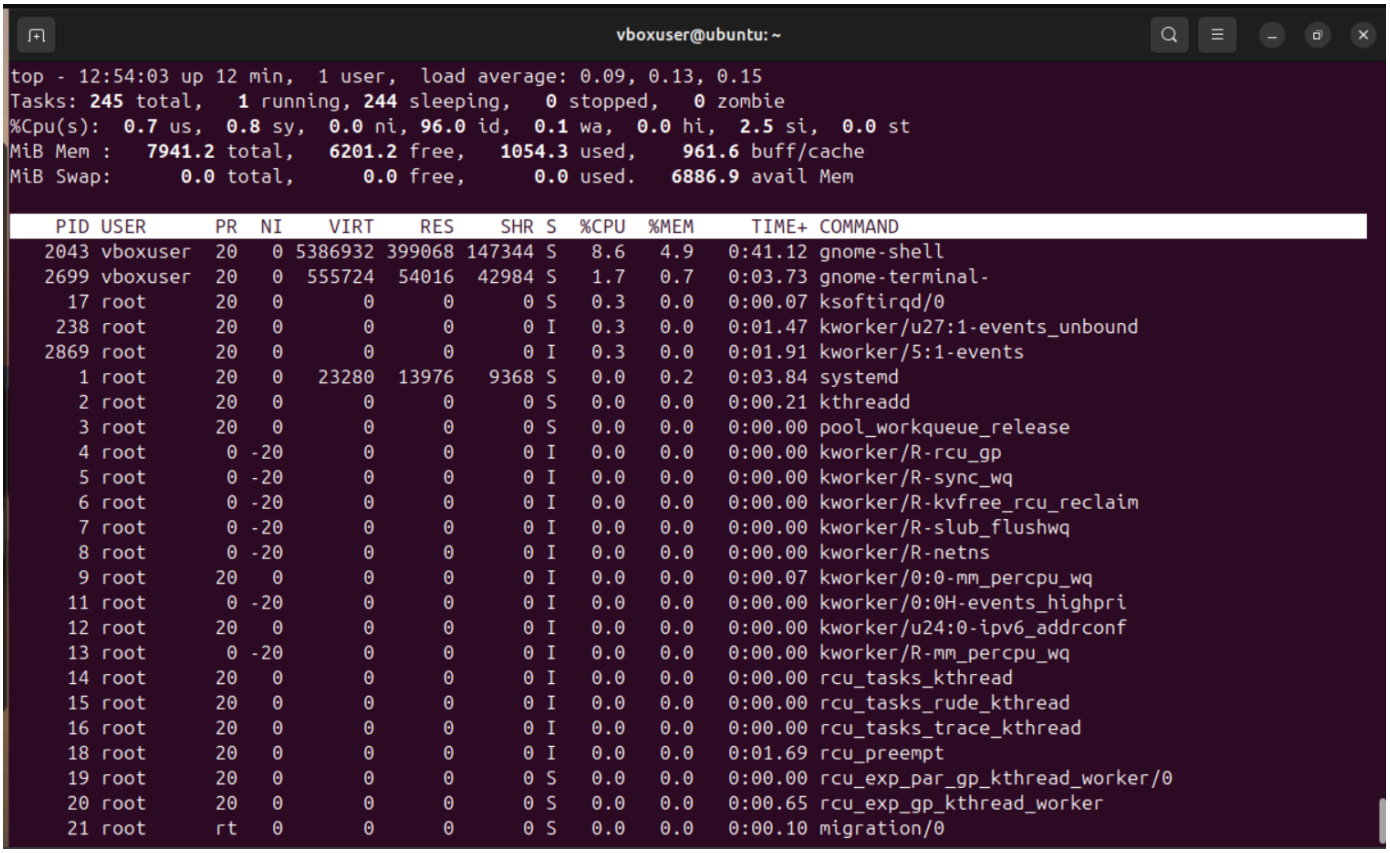
---

7. Real-Time Monitoring

Command:

```
top
```

Sample code:



Explanation:

Displays CPU, memory, and process usage in real-time.

- Press **q** to quit.

Example Output (partial):

```
top - 10:20:51 up 2 days,  3:12,  2 users,  load average: 0.22, 0.33, 0.45
Tasks: 197 total,   1 running, 196 sleeping,   0 stopped,   0 zombie
```

```
%Cpu(s): 12.3 us,  5.4 sy,  0.0 ni, 80.1 id,  2.2 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  8045632 total,  3564980 free,  1876324 used,  2604328 buff/cache
```

---

8. Adjust Process Priority

The `nice` command is used to start a process with a specified nice value, while the `renice` command is used to alter priority of the running process.
Syntax:
`nice -n <nice_value> <command_name>`

Value of `nice`: -20(highest) to +20(lowest).
Default: 0.

```
-n:
Set the nice value for a new process.
-p:
Modify the nice value for a process using its PID.
```

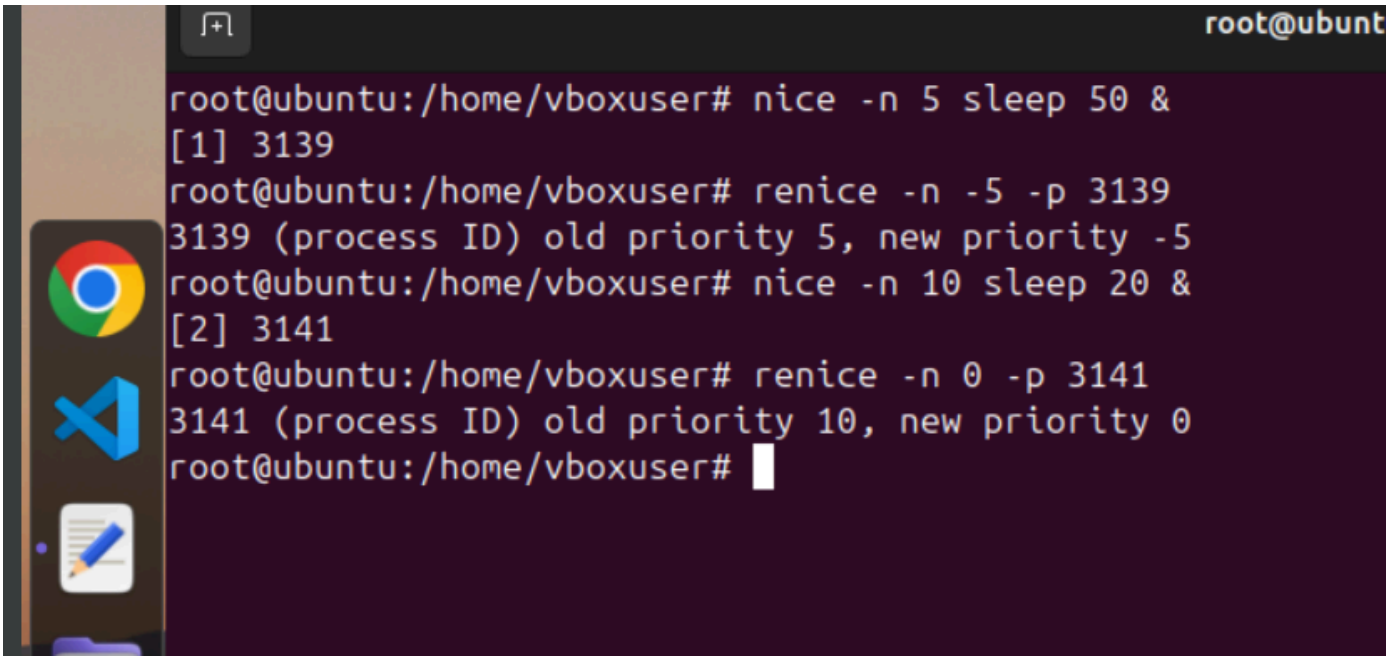
Start a process with low priority:

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

Output:

```
[1] 3050
```

Sample code:



Change priority of `running` process:

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

Note: You can only use renice when the process is running and hasnt ended.

Output:

```
3050 (process ID) old priority 10, new priority -5
```

> **Note:** Lower *nice* values (negative) mean **higher** priority. Requires appropriate privileges.

---

9. CPU Affinity (Bind Process to CPU Core)

Check CPU affinity:

```
taskset -cp 3050
```

Here, if u use -p instead of -cp then it just shows 3f (in my system)

Output:

```
pid 3050's current affinity list: 0-3
```

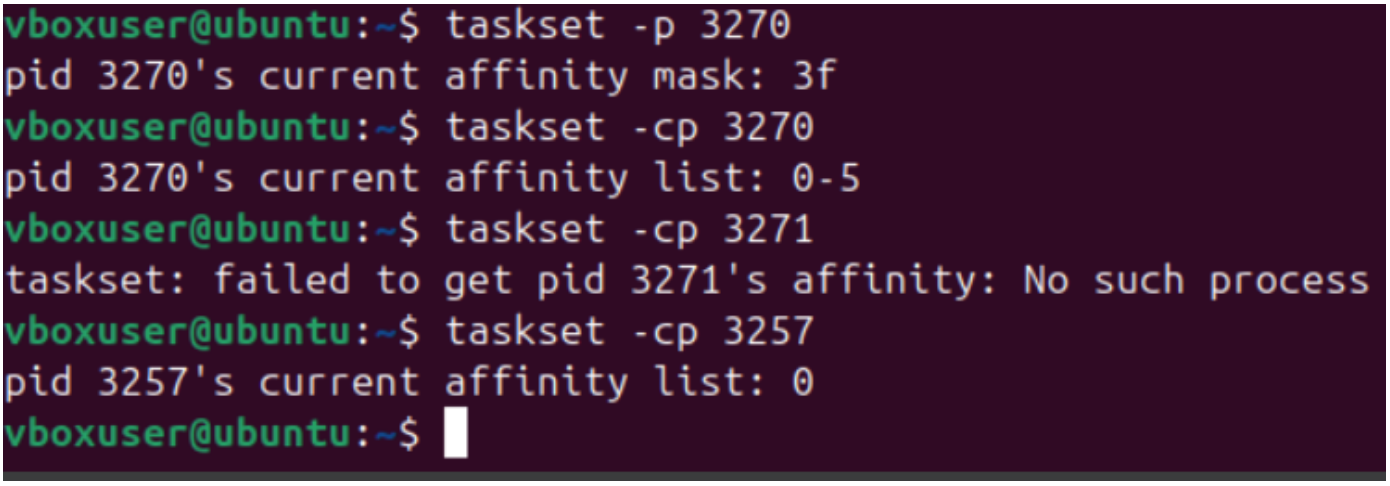
Restrict to core 1 only:

```
taskset -cp 1 3050
```

Output:

```
pid 3050's current affinity list: 1
```

Sample code:



10. I/O Scheduling Priority

Command:

Syntax:

`ionice -c scheduling_class -n priority_nice_value command`

Note: `ionice` affects Disk I/O not CPU.

Number of the scheduling classes:
-0 for none
-1 for real-time
-2 for best-exertion
-3 for inactive.

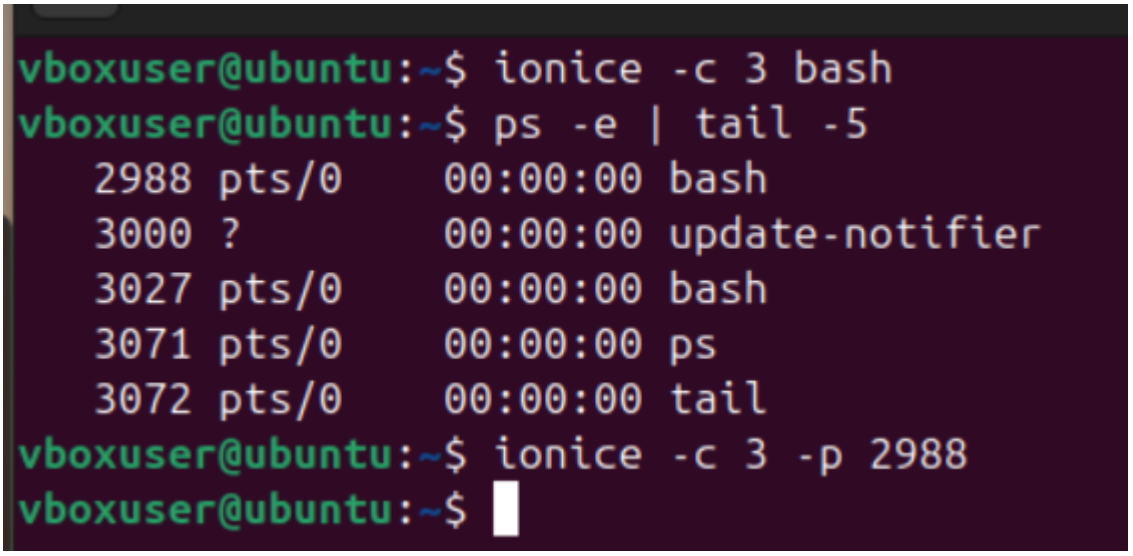
```
ionice -c 3 -p 3050
```

Output:

```
successfully set pid 3050's IO scheduling class to idle
```

- **Class 3 (idle)** → Process only gets I/O when system is idle.

Sample code:



---

11. File Descriptors Used by a Process

Command:

`lsof`: The command itself, used to list open files.
Note: This command lists out all the files that are opened by any process in the system.

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

Note:
`head -5` shows the first 5 lines of the output incl the column heads.

Example Output:

```
COMMAND  PID USER   FD   TYPE DEVICE SIZE/OFF   NODE NAME
sleep   3050 vibhu  cwd  DIR  253,0     4096  131073 /home/vibhu
sleep   3050 vibhu  rtd  DIR  253,0     4096       2 /
sleep   3050 vibhu  txt  REG  253,0    17520  133580 /usr/bin/sleep
```

Sample code:

```
vboxuser@ubuntu:~$ lsof -p 2988
COMMAND  PID     USER   FD   TYPE DEVICE SIZE/OFF    NODE NAME
bash    2988 vboxuser  cwd    DIR    8,2     4096 1179650 /home/vboxuser
bash    2988 vboxuser  rtd    DIR    8,2     4096       2 /
bash    2988 vboxuser  txt    REG    8,2  1446024 1311289 /usr/bin/bash
bash    2988 vboxuser  mem    REG    8,2  5719296 1368750 /usr/lib/locale/locale
-archive
bash    2988 vboxuser  mem    REG    8,2  2125328 1313198 /usr/lib/x86_64-linux-
gnu/libc.so.6
bash    2988 vboxuser  mem    REG    8,2   208328 1322742 /usr/lib/x86_64-linux-
gnu/libtinfo.so.6.4
bash    2988 vboxuser  mem    REG    8,2    27028 1313187 /usr/lib/x86_64-linux-
gnu/gconv/gconv-modules.cache
bash    2988 vboxuser  mem    REG    8,2   236616 1313195 /usr/lib/x86_64-linux-
gnu/ld-linux-x86-64.so.2
bash    2988 vboxuser    0u   CHR  136,0      0t0       3 /dev/pts/0
bash    2988 vboxuser    1u   CHR  136,0      0t0       3 /dev/pts/0
bash    2988 vboxuser    2u   CHR  136,0      0t0       3 /dev/pts/0
bash    2988 vboxuser  255u   CHR  136,0      0t0       3 /dev/pts/0
vboxuser@ubuntu:~$
```

---

12. Working with Strace Process Monitoring Tool

Command:

```
strace -p 3050
```

Example Output:

```
strace: Process 3050 attached
restart_syscall(<... resuming interrupted nanosleep ...>) = 0
nanosleep({tv_sec=300, tv_nsec=0}, 0x7ffd4a60d8b0) = ? ERESTART_RESTARTBLOCK
(Interrupted by signal)
```

> Press **Ctrl+C** to detach from `strace`.

Sample code:

```
vboxuser@ubuntu:~$ strace bash
execve("/usr/bin/bash", ["bash"], 0x7ffe4420f9c0 /* 48 vars */) = 0
brk(NULL)                               = 0x60fd172d3000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7d86dcf83000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=58027, ...}) = 0
mmap(NULL, 58027, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7d86dcf74000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libtinfo.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\0\0\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0644, st_size=208328, ...}) = 0
mmap(NULL, 211240, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7d86dcf40000
mmap(0x7d86dcf4e000, 77824, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0xe000) = 0x7d86dcf4e000
mmap(0x7d86dcf61000, 57344, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x21000) = 0x7d86dcf61000
mmap(0x7d86dcf6f000, 20480, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x2e000) = 0x7d86dcf6f000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\220\243\2\0\0\0\0\0"..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
fstat(3, {st_mode=S_IFREG|0755, st_size=2125328, ...}) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0"..., 784, 64) = 784
mmap(NULL, 2170256, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7d86dcc00000
mmap(0x7d86dcc28000, 1605632, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x28000) = 0x7d86dcc28000
mmap(0x7d86dcdb0000, 323584, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1b0000) = 0x7d86dcdb0000
mmap(0x7d86dcdff000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1fe000) = 0x7d86dcdff000
mmap(0x7d86dce05000, 52624, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7d86dce05000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7d86dcf3d000
arch_prctl(ARCH_SET_FS, 0x7d86dcf3d740) = 0
set_tid_address(0x7d86dcf3da10)         = 3102
set_robust_list(0x7d86dcf3da20, 24)     = 0
rseq(0x7d86dcf3e060, 0x20, 0, 0x53053053) = 0
mprotect(0x7d86dcdff000, 16384, PROT_READ) = 0
mprotect(0x7d86dcf6f000, 16384, PROT_READ) = 0
mprotect(0x60fd028e1000, 16384, PROT_READ) = 0
mprotect(0x7d86dcfc1000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0x7d86dcf74000, 58027)           = 0
openat(AT_FDCWD, "/dev/tty", O_RDWR|O_NONBLOCK) = 3
close(3)                                = 0
getrandom("\xc8\x84\x74\x62\x50\x42\x0e\xaf", 8, GRND_NONBLOCK) = 8
```

---

13. Find Process Using a Port

Command:

```
sudo fuser -n tcp 8080
#used to identify which process is using TCP port 8080
```

Example Output:

```
8080/tcp:             4321
```

> PID **4321** is using port **8080**.

NOTE: it will output null if no process is running in that port.

Sample code:

```
vboxuser@ubuntu:~$ sudo fuser -n tcp 1091
vboxuser@ubuntu:~$ sudo fuser -n tcp 0
0/tcp:                   570  1186
vboxuser@ubuntu:~$
```

14. Monitoring processes

Command:

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

- pidstat: A performance monitoring tool from the sysstat package.(not in my device ##)
- -p 3050: Monitor only the process with PID 3050.
- 2: Interval in seconds between each report.
- 3: Number of reports to generate.

Example Output:

```
Linux 5.15.0 (ubuntu)   09/25/25        _x86_64_        (4 CPU)
12:30:20      UID       PID    %usr %system  %CPU   CPU  Command
12:30:22     1000      3050    0.00    0.00   0.00     1  sleep
12:30:24     1000      3050    0.00    0.00   0.00     1  sleep
12:30:26     1000      3050    0.00    0.00   0.00     1  sleep
```

> **Tip:** `pidstat` is part of the `sysstat` package on most distros.

Sample code:

```
vboxuser@ubuntu:~$ pidstat -p 3113 3 3
Linux 6.14.0-27-generic (ubuntu)        09/26/2025      _x86_64_        (6 CPU)

02:56:24 PM   UID       PID    %usr %system  %guest   %wait    %CPU   CPU  Command
02:56:27 PM  1000      3113    0.00    0.00    0.00    0.00    0.00     2  bash
02:56:30 PM  1000      3113    0.00    0.00    0.00    0.00    0.00     2  bash
02:56:33 PM  1000      3113    0.00    0.00    0.00    0.00    0.00     2  bash
Average:     1000      3113    0.00    0.00    0.00    0.00    0.00     -  bash
vboxuser@ubuntu:~$
```

---

15. Control Groups for Resource Limits

`cgroups`

- a Linux kernel feature that limits, accounts for, and isolates the resource usage (CPU, memory, disk I/O, network, etc.) of a collection of processes.

Q 1: We will create a disk-controlled group so that we may run any process with a finite amount of disk read/writes available. i.e. we want to throttle reads and writes done by a process or a group of

processes.

Create a new cgroup:

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

Limit CPU and Memory:

```
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 a process (PID 3050) to cgroup:

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

> **Note:** Paths and controllers differ between **cgroups v1** and **v2**. Adjust for your system.

Sample code:

```
root@ubuntu:/# stat -fc %T /sys/fs/cgroup/
cgroup2fs
root@ubuntu:/# cd /sys/fs/cgroup
root@ubuntu:/sys/fs/cgroup# echo +cpu +memory | sudo tee cgroup.subtree_control
+cpu +memory
root@ubuntu:/sys/fs/cgroup# sudo mkdir firstgrp
root@ubuntu:/sys/fs/cgroup# ls -l firstgrp
total 0
-r--r--r-- 1 root root 0 Sep 26 15:17 cgroup.controllers
-r--r--r-- 1 root root 0 Sep 26 15:17 cgroup.events
-rw-r--r-- 1 root root 0 Sep 26 15:17 cgroup.freeze
--w------- 1 root root 0 Sep 26 15:17 cgroup.kill
-rw-r--r-- 1 root root 0 Sep 26 15:17 cgroup.max.depth
-rw-r--r-- 1 root root 0 Sep 26 15:17 cgroup.max.descendants
-rw-r--r-- 1 root root 0 Sep 26 15:17 cgroup.pressure
-rw-r--r-- 1 root root 0 Sep 26 15:17 cgroup.procs
-r--r--r-- 1 root root 0 Sep 26 15:17 cgroup.stat
-rw-r--r-- 1 root root 0 Sep 26 15:17 cgroup.subtree_control
-rw-r--r-- 1 root root 0 Sep 26 15:17 cgroup.threads
-rw-r--r-- 1 root root 0 Sep 26 15:17 cgroup.type
-rw-r--r-- 1 root root 0 Sep 26 15:17 cpu.idle
-rw-r--r-- 1 root root 0 Sep 26 15:17 cpu.max
-rw-r--r-- 1 root root 0 Sep 26 15:17 cpu.max.burst
-rw-r--r-- 1 root root 0 Sep 26 15:17 cpu.pressure
-r--r--r-- 1 root root 0 Sep 26 15:17 cpu.stat
-r--r--r-- 1 root root 0 Sep 26 15:17 cpu.stat.local
-rw-r--r-- 1 root root 0 Sep 26 15:17 cpu.uclamp.max
-rw-r--r-- 1 root root 0 Sep 26 15:17 cpu.uclamp.min
-rw-r--r-- 1 root root 0 Sep 26 15:17 cpu.weight
-rw-r--r-- 1 root root 0 Sep 26 15:17 cpu.weight.nice
-rw-r--r-- 1 root root 0 Sep 26 15:17 io.pressure
-r--r--r-- 1 root root 0 Sep 26 15:17 memory.current
-r--r--r-- 1 root root 0 Sep 26 15:17 memory.events
-r--r--r-- 1 root root 0 Sep 26 15:17 memory.events.local
-rw-r--r-- 1 root root 0 Sep 26 15:17 memory.high
-rw-r--r-- 1 root root 0 Sep 26 15:17 memory.low
-rw-r--r-- 1 root root 0 Sep 26 15:17 memory.max
-rw-r--r-- 1 root root 0 Sep 26 15:17 memory.min
-r--r--r-- 1 root root 0 Sep 26 15:17 memory.numa_stat
-rw-r--r-- 1 root root 0 Sep 26 15:17 memory.oom.group
-rw-r--r-- 1 root root 0 Sep 26 15:17 memory.peak
-rw-r--r-- 1 root root 0 Sep 26 15:17 memory.pressure
--w------- 1 root root 0 Sep 26 15:17 memory.reclaim
```

```
-rw-r--r-- 1 root root 0 Sep 26 15:17 memory.oom.group
-rw-r--r-- 1 root root 0 Sep 26 15:17 memory.peak
-rw-r--r-- 1 root root 0 Sep 26 15:17 memory.pressure
--w------- 1 root root 0 Sep 26 15:17 memory.reclaim
-r--r--r-- 1 root root 0 Sep 26 15:17 memory.stat
-r--r--r-- 1 root root 0 Sep 26 15:17 memory.swap.current
-r--r--r-- 1 root root 0 Sep 26 15:17 memory.swap.events
-rw-r--r-- 1 root root 0 Sep 26 15:17 memory.swap.high
-rw-r--r-- 1 root root 0 Sep 26 15:17 memory.swap.max
-rw-r--r-- 1 root root 0 Sep 26 15:17 memory.swap.peak
-r--r--r-- 1 root root 0 Sep 26 15:17 memory.zswap.current
-rw-r--r-- 1 root root 0 Sep 26 15:17 memory.zswap.max
-rw-r--r-- 1 root root 0 Sep 26 15:17 memory.zswap.writeback
-r--r--r-- 1 root root 0 Sep 26 15:17 pids.current
-r--r--r-- 1 root root 0 Sep 26 15:17 pids.events
-r--r--r-- 1 root root 0 Sep 26 15:17 pids.events.local
-rw-r--r-- 1 root root 0 Sep 26 15:17 pids.max
-r--r--r-- 1 root root 0 Sep 26 15:17 pids.peak
root@ubuntu:/sys/fs/cgroup# echo $$ | sudo tee /firstgrp/cgroup.procs
tee: /firstgrp/cgroup.procs: No such file or directory
3149
root@ubuntu:/sys/fs/cgroup# cd firstgrp
root@ubuntu:/sys/fs/cgroup/firstgrp# echo $$ | sudo tee cgroup.procs
3149
root@ubuntu:/sys/fs/cgroup/firstgrp# echo "50000 100000" | sudo tee cpu.max
50000 100000
root@ubuntu:/sys/fs/cgroup/firstgrp# echo 104857600 | sudo tee memory.max
104857600
root@ubuntu:/sys/fs/cgroup/firstgrp# cat cpu.stat
usage_usec 185642
user_usec 59247
system_usec 126394
nice_usec 0
core_sched.force_idle_usec 0
nr_periods 16
nr_throttled 0
throttled_usec 0
nr_bursts 0
burst_usec 0
root@ubuntu:/sys/fs/cgroup/firstgrp# cat memory.current
675840
root@ubuntu:/sys/fs/cgroup/firstgrp#
```

---

16. Alternatives to nice / renice

1. `chrt` command

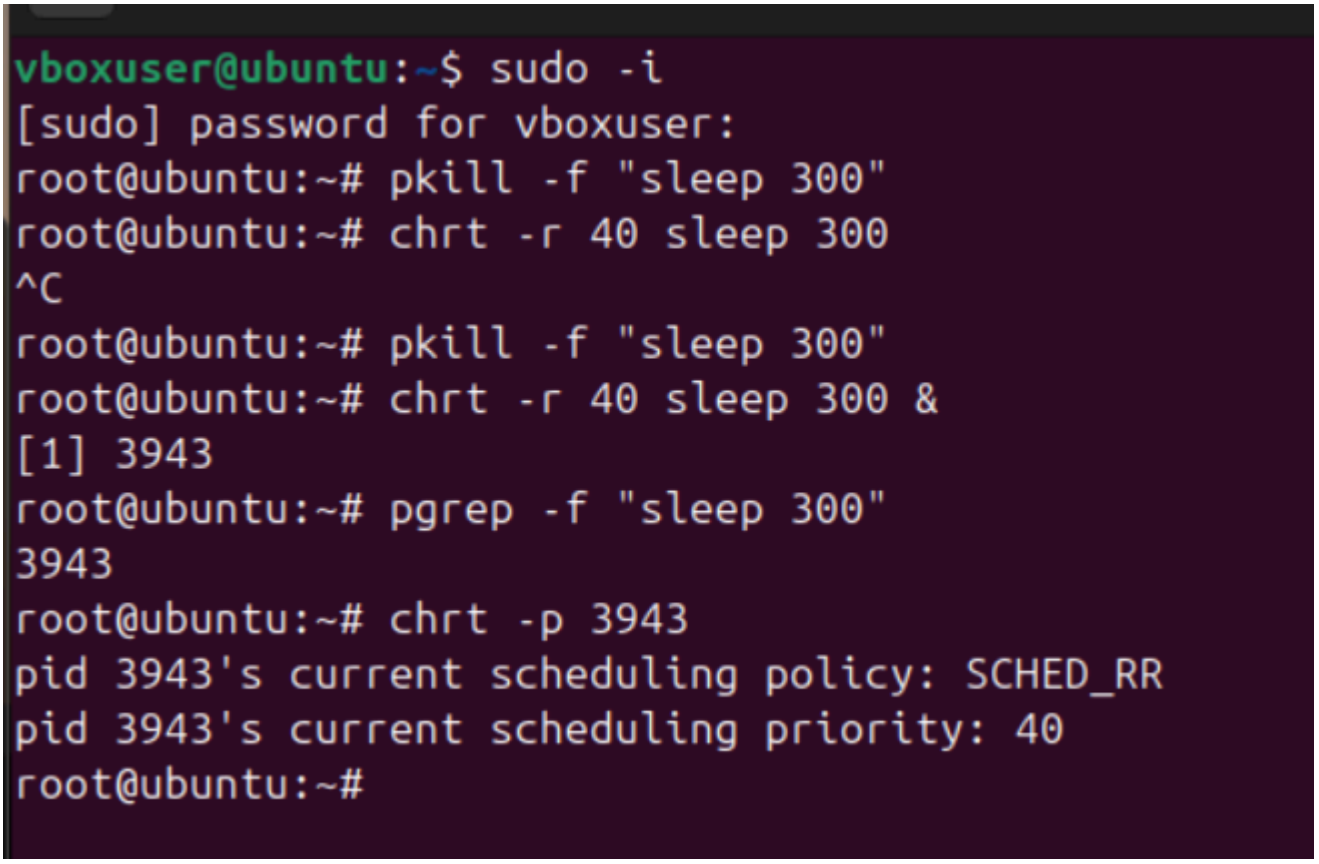
- command-line utility used to set or retrieve the real-time scheduling policy and priority of a process in Linux.

🧭 Linux Scheduling Policies: `SCHED_FIFO`, `SCHED_RR`, `SCHED_OTHER`

Policy	Type	Priority Range	Time Slice	Preemption	Use Case
`SCHED_FIFO`	Real-time	1–99	None	Yes	Deterministic tasks needing strict order
`SCHED_RR`	Real-time	1–99	Yes	Yes	Fair time-sharing among real-time tasks
`SCHED_OTHER`	Normal (default)	0	Yes	Yes	General-purpose user processes

Notes:

- `SCHED_FIFO`: Runs until blocked or preempted. No time slice.
  ⛳FLAG: `-f`
- `SCHED_RR`: Round-robin with time slices. Fair among equal-priority tasks.
  ⛳ FLAG : `-r`
- `SCHED_OTHER`: Default Linux scheduler (CFS). Used by most user-space processes.
  ⛳ FLAG: `-o`



2. ionice (I/O Priority Control)
   Refer to: [I/O Scheduling Priority]

3. taskset (CPU Affinity)
   Refer to : CPU Affinity (Bind Process to CPU Core)
4. Control Groups (cgroups)
   Refer to : Control Groups for Resource Limits
5. systemd-run
   Code:
   `systemd-run --scope -p CPUWeight=200 stress --cpu 4`

Output:
launching the stress tool inside a transient systemd scope unit with a CPU weight of 200.

Explaination:

- `systemd-run`: Creates and starts a transient systemd unit.
- `--scope`: Runs the command in a scope unit, which is suitable for processes that are not started by systemd itself (like interactive commands).
  -`-p CPUWeight=200`: Sets the CPU weight for the scope unit. This is part of systemd's CPU scheduling via cgroups. The default weight is 100; higher values give more CPU time relative to other units.
- `stress --cpu 4`: Runs the stress tool to spawn 4 CPU workers, each spinning in a tight loop to simulate CPU load.

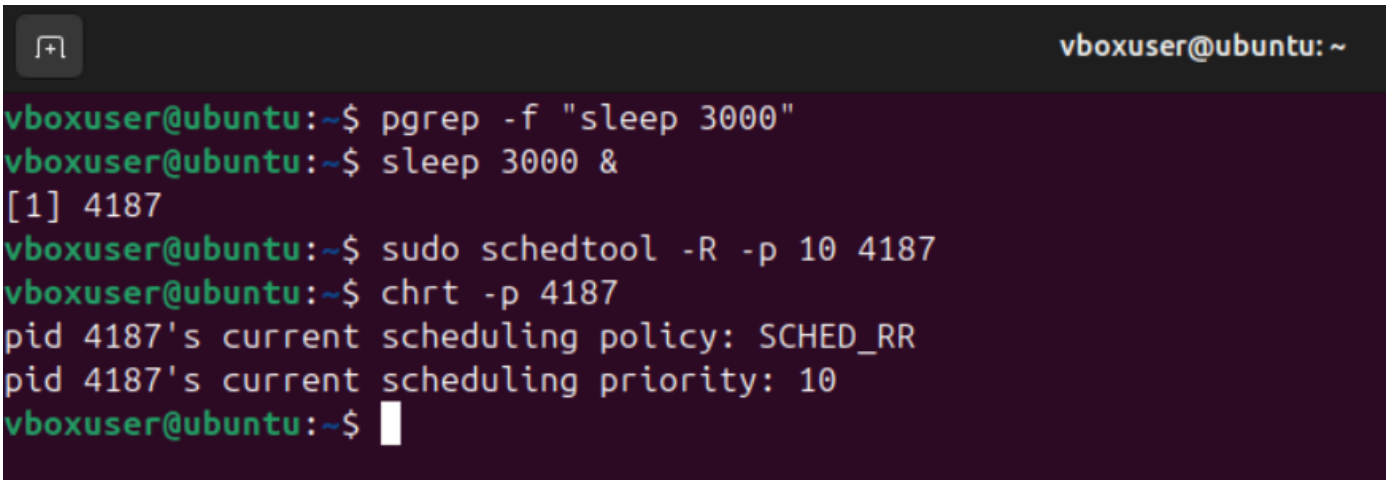
```
vboxuser@ubuntu:~$ sudo apt install stress
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  libllvm19
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  stress
0 upgraded, 1 newly installed, 0 to remove and 36 not upgraded.
Need to get 18.1 kB of archives.
After this operation, 52.2 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu noble/universe amd64 stress amd64 1.0.7-1 [18.1 kB]
Fetched 18.1 kB in 1s (29.0 kB/s)
Selecting previously unselected package stress.
(Reading database ... 202873 files and directories currently installed.)
Preparing to unpack .../stress_1.0.7-1_amd64.deb ...
Unpacking stress (1.0.7-1) ...
Setting up stress (1.0.7-1) ...
Processing triggers for man-db (2.12.0-4build2) ...
vboxuser@ubuntu:~$ systemd-run --scope -p CPUWeight=200 stress --cpu 4
Running as unit: run-u111.scope; invocation ID: ceea114cb00f48d1909ed1d31ab24431
stress: info: [3641] dispatching hogs: 4 cpu, 0 io, 0 vm, 0 hdd
^C
vboxuser@ubuntu:~$ top

top - 06:11:52 up 15 min,  1 user,  load average: 0.86, 0.82, 0.56
Tasks: 248 total,   1 running, 247 sleeping,   0 stopped,   0 zombie
```

7. schedtool
   Syntax:

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

Sample code:



17. 📌 Cheat Sheet

Command	Purpose
`ps -C processname`	Filter processes by name
`ps -p PID`	Show details for specific PID
`ps -u username`	List processes by user
`ps -C gedit -L`	List threads of a process
`ps -e -H`	Show child processes in hierarchy
`ps -C gedit -o pid,tty`	Customize output columns
`strace -p PID`	Monitor system calls of a process
`sudo fuser -n tcp PORT`	Find process using a TCP port
`pidstat -p PID interval count`	Monitor CPU usage over time
`cgcreate -g cpu,memory:/group`	Create a control group
`echo PID | tee /sys/fs/cgroup/...`	Add process to cgroup
`systemd-run --scope -p CPUWeight=200 command`	Run process with CPU weight control
`schedtool -R -p N PID`	Set real-time scheduling policy