good morning dear sir and freinds
in todays session i am going to practice the essential monitring and mangaing process

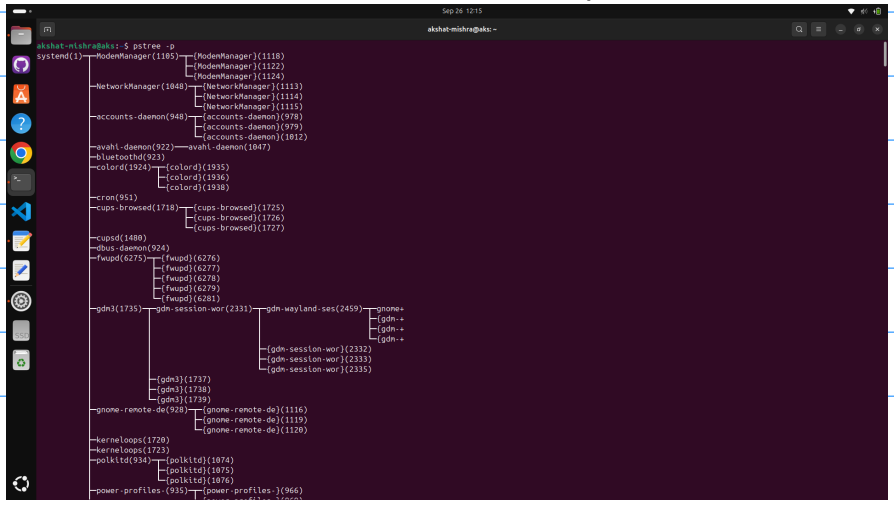
1. (ps aux) Explanation:
a → show processes for all users
u → show user/owner of process
x → show processes not attached to a terminal

so in the first command we have got ps aux here is the output of that command

```
akshat-mishra@aks:~$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0  23832 14352 ?        Ss   10:45   0:03 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    10:45   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        S    10:45   0:00 [pool_workqueue_release]
root           4  0.0  0.0      0     0 ?        I<   10:45   0:00 [kworker/R-rcu_gp]
root           5  0.0  0.0      0     0 ?        I<   10:45   0:00 [kworker/R-sync_wq]
root           6  0.0  0.0      0     0 ?        I<   10:45   0:00 [kworker/R-kvfree_rcu_reclaim]
root           7  0.0  0.0      0     0 ?        I<   10:45   0:00 [kworker/R-slub_flushwq]
root           8  0.0  0.0      0     0 ?        I<   10:45   0:00 [kworker/R-netns]
root          10  0.0  0.0      0     0 ?        I    10:45   0:00 [kworker/0:1-events]
root          11  0.0  0.0      0     0 ?        I<   10:45   0:00 [kworker/0:0H-events_highpri]
root          13  0.0  0.0      0     0 ?        I<   10:45   0:00 [kworker/R-mm_percpu_wq]
root          14  0.0  0.0      0     0 ?        I    10:45   0:00 [rcu_tasks_kthread]
root          15  0.0  0.0      0     0 ?        I    10:45   0:00 [rcu_tasks_rude_kthread]
root          16  0.0  0.0      0     0 ?        I    10:45   0:00 [rcu_tasks_trace_kthread]
root          17  0.0  0.0      0     0 ?        S    10:45   0:00 [ksoftirqd/0]
root          18  0.1  0.0      0     0 ?        I    10:45   0:19 [rcu_preempt]
root          19  0.0  0.0      0     0 ?        S    10:45   0:00 [rcu_exp_par_gp_kthread_worker/0]
root          20  0.0  0.0      0     0 ?        S    10:45   0:00 [rcu_exp_gp_kthread_worker]
root          21  0.0  0.0      0     0 ?        S    10:45   0:01 [migration/0]
```

in this command it shows pid , cpu, mem and many more used by the system to complete for process and also define who is the user

2.(pstree)
Process Tree Command: Plain Text pstree -p
The -p option with the pstree command in Linux serves to display the Process IDs (PIDs) alongside each process name in the hierarchical tree output.



3. (top)

Real-Time Monitoring Command: Plain Text top

Top's first line, top, shows the same information
as the uptime command. The first value is the system time.
The second value represents how long the system has been up and running,
while the third value indicates the current number of users on the system.
The final values are the load average for the system

output:-

```
top - 19:45:50 up 6 min,  1 user,  load average: 0.52, 0.52, 0.26
Tasks: 330 total,   1 running, 329 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.4 us,  0.4 sy,  0.0 ni, 99.0 id,  0.1 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  15240.9 total,  12828.8 free,   1565.3 used,   1173.3 buff/cache
MiB Swap:   4096.0 total,   4096.0 free,      0.0 used.  13675.6 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   2611 akshat-+  20   0 5338072 372304 177008 S   4.0   2.4   0:34.45 gnome-shell
   3912 akshat-+  20   0  559904  58256  47164 S   3.0   0.4   0:00.48 gnome-terminal-
    239 root     -51   0       0      0      0 S   1.3   0.0   0:03.31 irq/55-ELAN07FB:00
    104 root     -51   0       0      0      0 S   0.7   0.0   0:02.38 irq/9-acpi
    509 root      20   0       0      0      0 I   0.7   0.0   0:02.31 kworker/u48:13-events_power_efficient
    840 root     -51   0       0      0      0 S   0.7   0.0   0:02.02 irq/95-rtw89_pci
   3931 akshat-+  20   0   14536   5908   3732 R   0.7   0.0   0:00.06 top
    126 root      20   0       0      0      0 S   0.3   0.0   0:00.01 hwrng
    143 root      20   0       0      0      0 I   0.3   0.0   0:00.17 kworker/5:2-events
    160 root      20   0       0      0      0 I   0.3   0.0   0:02.06 kworker/u48:3-phy0
    642 root      20   0       0      0      0 I   0.3   0.0   0:00.19 kworker/4:2-events
      1 root      20   0   23628  14068   9204 S   0.0   0.1   0:01.52 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 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
     10 root      20   0       0      0      0 I   0.0   0.0   0:00.04 kworker/0:1-events
     11 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
     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   0:00.00 ksoftirqd/0
     18 root      20   0       0      0      0 I   0.0   0.0   0:00.71 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   0:00.00 rcu_exp_gp_kthread_worker
     21 root      rt   0       0      0      0 S   0.0   0.0   0:00.03 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   0:00.23 migration/1
     27 root      20   0       0      0      0 S   0.0   0.0   0:00.01 ksoftirqd/1
     28 root      20   0       0      0      0 I   0.0   0.0   0:00.29 kworker/1:0-mm_percpu_wq
     29 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/1:0H-events_highpri
```

4. (nice and renice)

Adjust Process Priority Start a process with low priority:

Plain Text nice -n 10 sleep 300 &Change priority of running process:

Plain Text renice -n -5 -p 3050

The nice and renice commands in Linux are used to manage the scheduling priority of processes, also known as "niceness." Niceness values range from -20 (highest priority) to 19 (lowest priority).

output:-

```
akshakshat-mishra@aks:~$ renice -n-5-p 3050
renirenice: invalid priority '-n-5-p'
Try Try 'renice --help' for more information.
akshakshat-mishra@aks:~$  nice -n 10 hello.sh
nicenice: ‘hello.sh’: No such file or directory
akshakshat-mishra@aks:~$ touch hello.sh
akshakshat-mishra@aks:~$ nice -n 10 hello.sh
nicenice: ‘hello.sh’: No such file or directory
```

5. (tasket)

CPU Affinity (Bind Process to CPU Core) Command:

Plain Text taskset -cp 3050 Example

pid 3050's current affinity list: 1

tasket -cp1 3050

The taskset command in Linux is used to manage the CPU affinity of processes. CPU affinity is a scheduler property that "bonds" a process to a specific set of CPUs or CPU cores on a system.

output:-

```
akshat-mishra@aks:~$ taskset -cp 3050
pid 3050's current affinity list: 1
akshat-mishra@aks:~$ taskset -cp 1 3050
pid 3050's current affinity list: 1
pid 3050's new affinity list: 1
```

6. (ionnice)
I/O Scheduling Priority Command: Plain Text ionice -c 3 -p 3050
onice is a command line-based utility in Linux-based
operating systems that helps in encouraging
the director to dole out needs on a process cycle level
to accelerate or to confine the disk subsystem
use of a particular cycle.

7. File Descriptors Used by a Process Command: Plain Text lsof -p 3050 | head -5
thios command we ue is lsof
The lsof command in Linux stands for "List Open Files."
It is a powerful command-line utility used to display information about files
that are currently open by processes on the system. In Unix-like operating system
s, including Linux, the concept of "everything is a file" extends beyond regular files
to include directories, block devices, character devices, libraries, executables,
named pipes, network sockets, and more.
lsof can provide details about all these open "files."

output:-

```
akshat-mishra@aks:~$ lsof -p 3050 | head -5
```

8.(strace)
Trace System Calls of a Process Command: Plain Text strace -p 3050
he strace command in Linux is a powerful utility used for
tracing system calls and signals made by a process.
It provides a detailed log of the interactions between a user
-space program and the Linux kernel, making it an invaluable tool for
debugging, troubleshooting, and understanding program behavior.

```
akshat-mishra@aks:~$ sudo strace -p 3050
[sudo] password for akshat-mishra:
strace: Process 3050 attached
futex(0x62c8f44120a0, FUTEX_WAIT_PRIVATE, 1, NULL
```

9.(fuser) Find Process Using a Port Command: Plain Text sudo fuser -n tcp 8080
The fuser command in Linux is a utility used to identify processes that are currently usin
Identifying processes using a file or directory:
When you need to unmount a filesystem, delete a file, or
modify a directory,
you might encounter errors if a process is still actively using it
. fuser helps you pinpoint which processes are holding those resources.

output:-

```
akshat-mishra@aks:~$ sudo fuser -n tcp 8080
akshat-mishra@aks:~$
```

10. (pidstat)Per-Process Statistics Command: Plain Text pidstat -p 3050 2 3
The pidstat command in Linux is a powerful utility used for
monitoring and reporting statistics for
individual tasks (processes and kernel threads)
managed by the Linux kernel. It is part of the sysstat package,
which provides various tools for system performance monitoring.

output :-

```
akshat-mishra@aks:~$ pidstat -p 3050 2 3
Linux 6.14.0-28-generic (aks)   09/26/2025      _x86_64_        (12 CPU)

12:49:53 PM   UID       PID    %usr %system  %guest   %wait    %CPU   CPU  Command
12:49:55 PM  1000      3050    0.00    0.00    0.00    0.00    0.00     1  pool-spawner
12:49:57 PM  1000      3050    0.00    0.00    0.00    0.00    0.00     1  pool-spawner
12:49:59 PM  1000      3050    0.00    0.00    0.00    0.00    0.00     1  pool-spawner
Average:     1000      3050    0.00    0.00    0.00    0.00    0.00     -  pool-spawner
akshat-mishra@aks:~$
```

11. cgroups — Control Resource Limits

Control Groups (cgroups) are a Linux kernel feature that lets you limit, prioritize,

Think of cgroups as a way to say:

“This group of processes can only use this much CPU and memory.”

output :-

```
akshat-mishra@aks:~$ sudo cgcreate -g cpu,memory:/testgroup
akshat-mishra@aks:~$ 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
akshat-mishra@aks:~$ echo 100M | sudo tee /sys/fs/cgroup/memory/testgroup/memory.limit_in_bytes
tee: /sys/fs/cgroup/memory/testgroup/memory.limit_in_bytes: No such file or directory
100M
akshat-mishra@aks:~$ echo 3050 | sudo tee /sys/fs/cgroup/cpu/testgroup/cgroup.procs
tee: /sys/fs/cgroup/cpu/testgroup/cgroup.procs: No such file or directory
3050
akshat-mishra@aks:~$ cat /sys/fs/cgroup/cpu/testgroup/cpuacct.usage
cat /sys/fs/cgroup/memory/testgroup/memory.usage_in_bytes
cat: /sys/fs/cgroup/cpu/testgroup/cpuacct.usage: No such file or directory
cat: /sys/fs/cgroup/memory/testgroup/memory.usage_in_bytes: No such file or directory
akshat-mishra@aks:~$
```