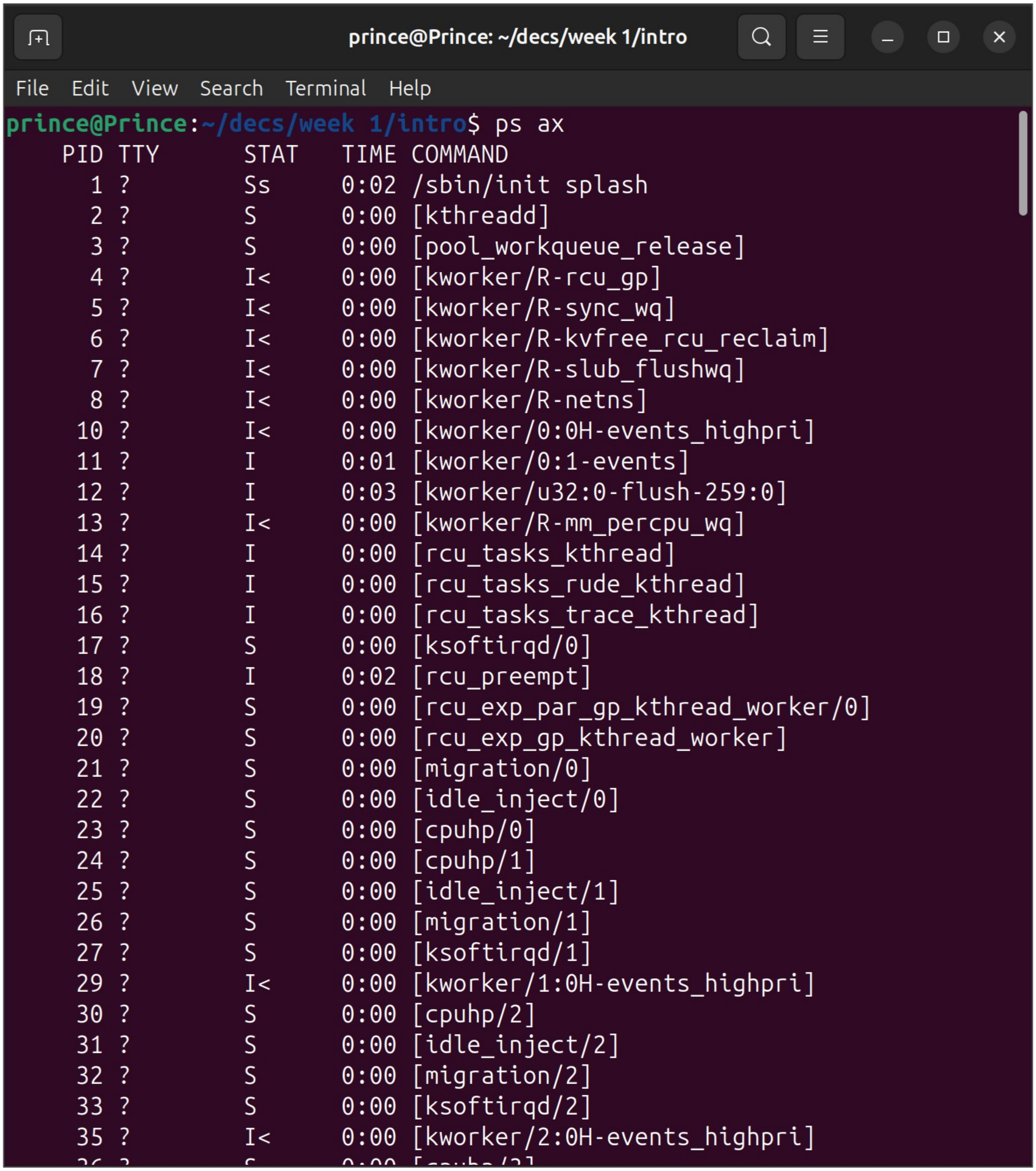
Prince
25M0809

EXERCISES -

1. We will begin with Linux shell commands used to examine the processes running in a system.
   (a) List all processes running in the system. Explore the different styles of displaying the output, e.g., BSD syntax vs. standard (ps) syntax.

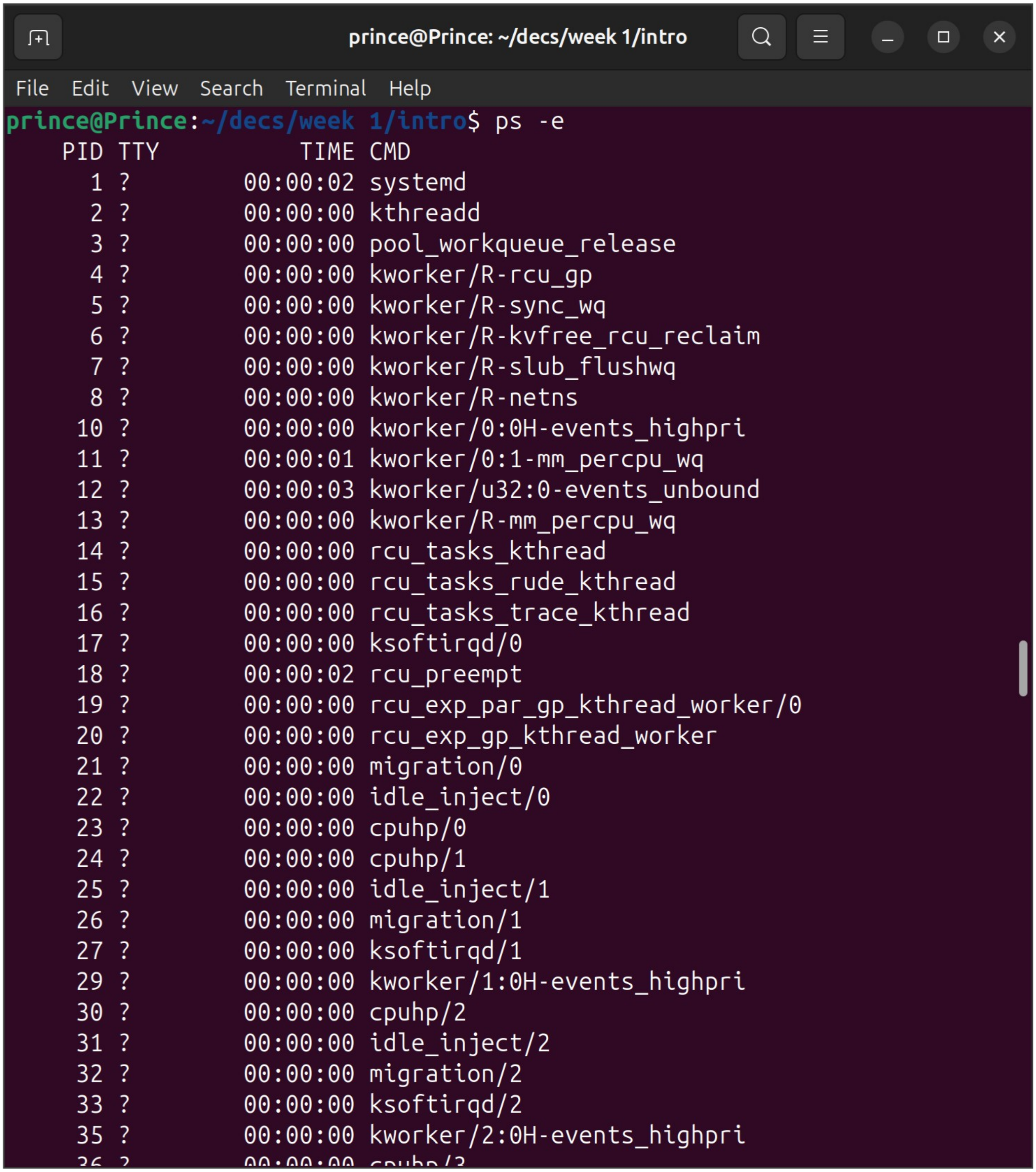
ANS →

BSD Syntax → ps ax



```
prince@Prince:~/decs/week 1/intro$ ps ax
    PID TTY      STAT   TIME COMMAND
      1 ?        Ss     0:02 /sbin/init splash
      2 ?        S      0:00 [kthreadd]
      3 ?        S      0:00 [pool_workqueue_release]
      4 ?        I<     0:00 [kworker/R-rcu_gp]
      5 ?        I<     0:00 [kworker/R-sync_wq]
      6 ?        I<     0:00 [kworker/R-kvfree_rcu_reclaim]
      7 ?        I<     0:00 [kworker/R-slub_flushwq]
      8 ?        I<     0:00 [kworker/R-netns]
     10 ?        I<     0:00 [kworker/0:0H-events_highpri]
     11 ?        I      0:01 [kworker/0:1-events]
     12 ?        I      0:03 [kworker/u32:0-flush-259:0]
     13 ?        I<     0:00 [kworker/R-mm_percpu_wq]
     14 ?        I      0:00 [rcu_tasks_kthread]
     15 ?        I      0:00 [rcu_tasks_rude_kthread]
     16 ?        I      0:00 [rcu_tasks_trace_kthread]
     17 ?        S      0:00 [ksoftirqd/0]
     18 ?        I      0:02 [rcu_preempt]
     19 ?        S      0:00 [rcu_exp_par_gp_kthread_worker/0]
     20 ?        S      0:00 [rcu_exp_gp_kthread_worker]
     21 ?        S      0:00 [migration/0]
     22 ?        S      0:00 [idle_inject/0]
     23 ?        S      0:00 [cpuhp/0]
     24 ?        S      0:00 [cpuhp/1]
     25 ?        S      0:00 [idle_inject/1]
     26 ?        S      0:00 [migration/1]
     27 ?        S      0:00 [ksoftirqd/1]
     29 ?        I<     0:00 [kworker/1:0H-events_highpri]
     30 ?        S      0:00 [cpuhp/2]
     31 ?        S      0:00 [idle_inject/2]
     32 ?        S      0:00 [migration/2]
     33 ?        S      0:00 [ksoftirqd/2]
     35 ?        I<     0:00 [kworker/2:0H-events_highpri]
     36 ?        S      0:00 [cpuhp/3]
```

Standard Syntax → ps -e

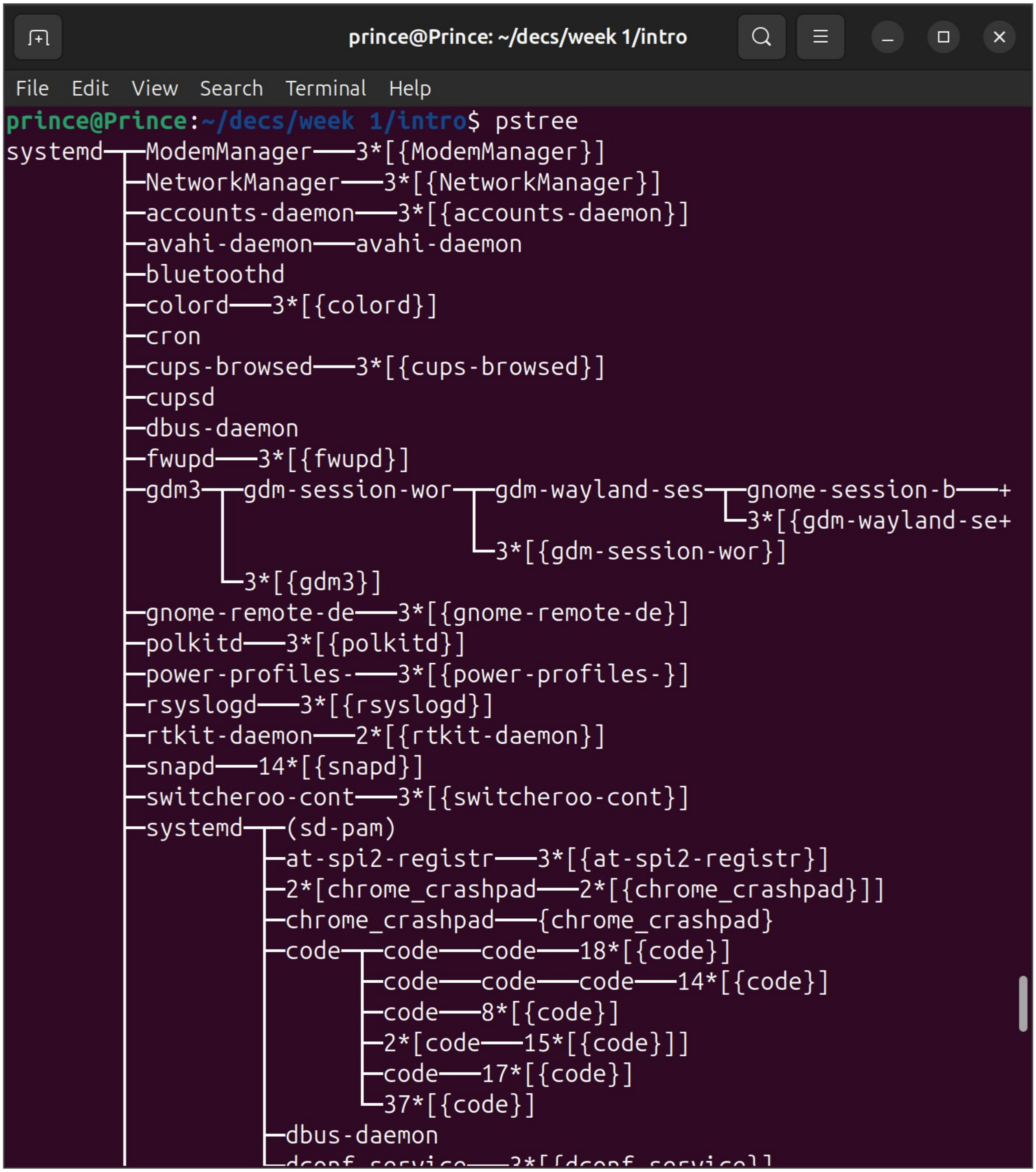


```
prince@Prince:~/decs/week 1/intro$ ps -e
    PID TTY          TIME CMD
      1 ?        00:00:02 systemd
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00 pool_workqueue_release
      4 ?        00:00:00 kworker/R-rcu_gp
      5 ?        00:00:00 kworker/R-sync_wq
      6 ?        00:00:00 kworker/R-kvfree_rcu_reclaim
      7 ?        00:00:00 kworker/R-slub_flushwq
      8 ?        00:00:00 kworker/R-netns
     10 ?        00:00:00 kworker/0:0H-events_highpri
     11 ?        00:00:01 kworker/0:1-mm_percpu_wq
     12 ?        00:00:03 kworker/u32:0-events_unbound
     13 ?        00:00:00 kworker/R-mm_percpu_wq
     14 ?        00:00:00 rcu_tasks_kthread
     15 ?        00:00:00 rcu_tasks_rude_kthread
     16 ?        00:00:00 rcu_tasks_trace_kthread
     17 ?        00:00:00 ksoftirqd/0
     18 ?        00:00:02 rcu_preempt
     19 ?        00:00:00 rcu_exp_par_gp_kthread_worker/0
     20 ?        00:00:00 rcu_exp_gp_kthread_worker
     21 ?        00:00:00 migration/0
     22 ?        00:00:00 idle_inject/0
     23 ?        00:00:00 cpuhp/0
     24 ?        00:00:00 cpuhp/1
     25 ?        00:00:00 idle_inject/1
     26 ?        00:00:00 migration/1
     27 ?        00:00:00 ksoftirqd/1
     29 ?        00:00:00 kworker/1:0H-events_highpri
     30 ?        00:00:00 cpuhp/2
     31 ?        00:00:00 idle_inject/2
     32 ?        00:00:00 migration/2
     33 ?        00:00:00 ksoftirqd/2
     35 ?        00:00:00 kworker/2:0H-events_highpri
     36 ?        00:00:00 cpuhp/3
```

(b) Print a process tree for all processes running in the system.

ANS →

Command → pstree

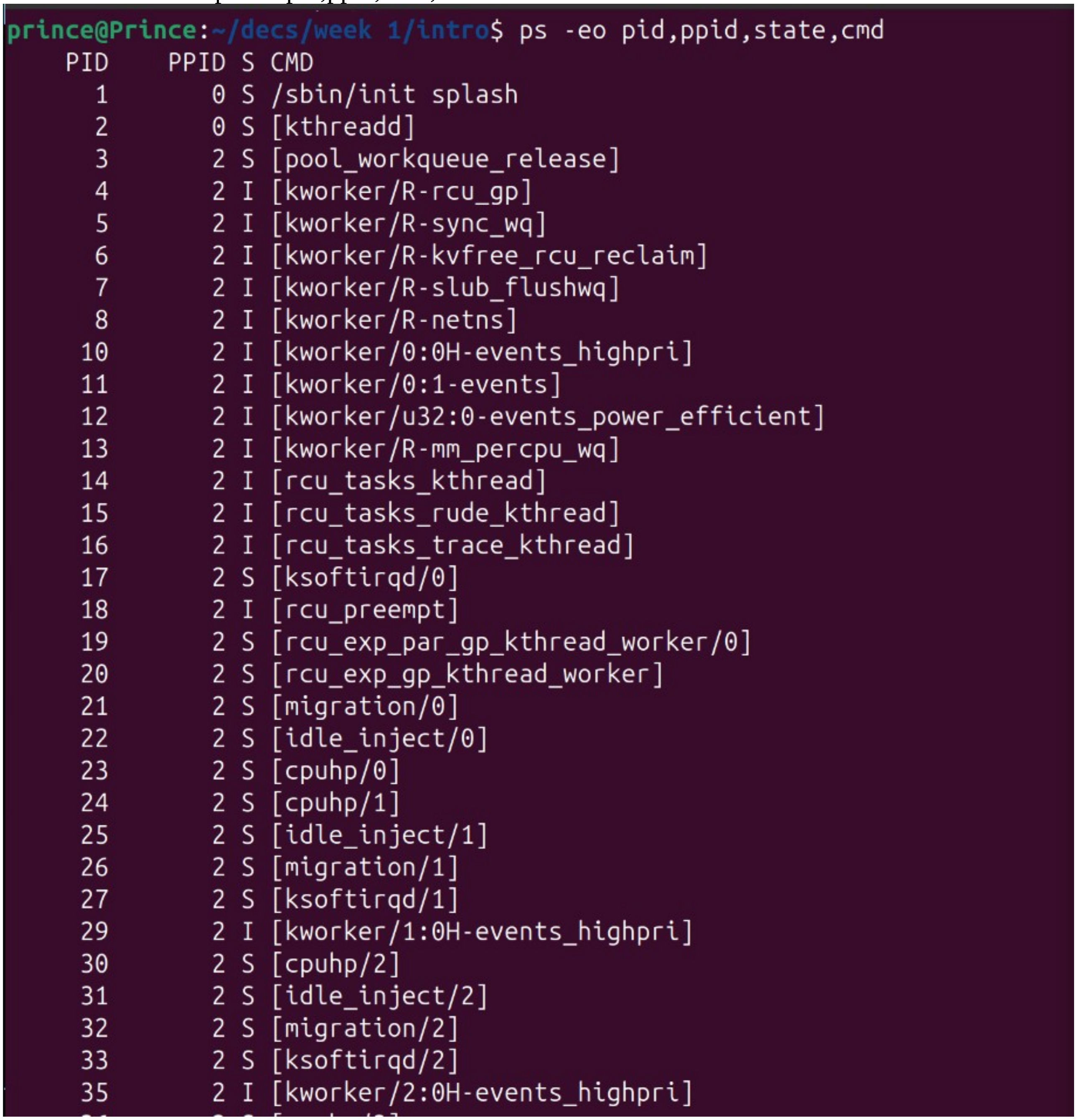


```
prince@Prince:~/decs/week 1/intro$ pstree
systemd─┬─ModemManager───3*[{ModemManager}]
        ├─NetworkManager───3*[{NetworkManager}]
        ├─accounts-daemon───3*[{accounts-daemon}]
        ├─avahi-daemon───avahi-daemon
        ├─bluetoothd
        ├─colord───3*[{colord}]
        ├─cron
        ├─cups-browsed───3*[{cups-browsed}]
        ├─cupsd
        ├─dbus-daemon
        ├─fwupd───3*[{fwupd}]
        ├─gdm3─┬─gdm-session-wor─┬─gdm-wayland-ses─┬─gnome-session-b───+
        │      │                 │                 └─3*[{gdm-wayland-se+
        │      │                 └─3*[{gdm-session-wor}]
        │      └─3*[{gdm3}]
        ├─gnome-remote-de───3*[{gnome-remote-de}]
        ├─polkitd───3*[{polkitd}]
        ├─power-profiles-───3*[{power-profiles-}]
        ├─rsyslogd───3*[{rsyslogd}]
        ├─rtkit-daemon───2*[{rtkit-daemon}]
        ├─snapd───14*[{snapd}]
        ├─switcheroo-cont───3*[{switcheroo-cont}]
        ├─systemd─┬─(sd-pam)
        │         ├─at-spi2-registr───3*[{at-spi2-registr}]
        │         ├─2*[chrome_crashpad───2*[{chrome_crashpad}]]
        │         ├─chrome_crashpad───{chrome_crashpad}
        │         ├─code─┬─code───code───18*[{code}]
        │         │      ├─code───code───code───14*[{code}]
        │         │      ├─code───8*[{code}]
        │         │      ├─2*[code───15*[{code}]]
        │         │      ├─code───17*[{code}]
        │         │      └─37*[{code}]
        │         ├─dbus-daemon
        │         └─dconf-service───3*[{dconf-service}]
```

(c) List the pid, ppid, state, command, for all process running in the system.

ANS →

Command → ps -eo pid,ppid,state,cmd

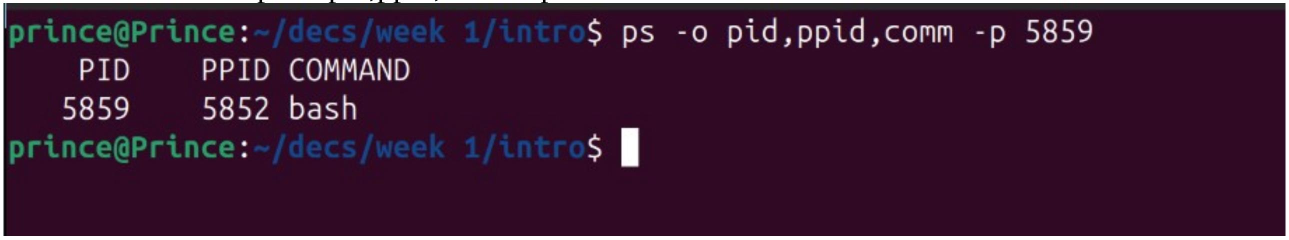


```
prince@Prince:~/decs/week 1/intro$ ps -eo pid,ppid,state,cmd
    PID    PPID S CMD
      1       0 S /sbin/init splash
      2       0 S [kthreadd]
      3       2 S [pool_workqueue_release]
      4       2 I [kworker/R-rcu_gp]
      5       2 I [kworker/R-sync_wq]
      6       2 I [kworker/R-kvfree_rcu_reclaim]
      7       2 I [kworker/R-slub_flushwq]
      8       2 I [kworker/R-netns]
     10       2 I [kworker/0:0H-events_highpri]
     11       2 I [kworker/0:1-events]
     12       2 I [kworker/u32:0-events_power_efficient]
     13       2 I [kworker/R-mm_percpu_wq]
     14       2 I [rcu_tasks_kthread]
     15       2 I [rcu_tasks_rude_kthread]
     16       2 I [rcu_tasks_trace_kthread]
     17       2 S [ksoftirqd/0]
     18       2 I [rcu_preempt]
     19       2 S [rcu_exp_par_gp_kthread_worker/0]
     20       2 S [rcu_exp_gp_kthread_worker]
     21       2 S [migration/0]
     22       2 S [idle_inject/0]
     23       2 S [cpuhp/0]
     24       2 S [cpuhp/1]
     25       2 S [idle_inject/1]
     26       2 S [migration/1]
     27       2 S [ksoftirqd/1]
     29       2 I [kworker/1:0H-events_highpri]
     30       2 S [cpuhp/2]
     31       2 S [idle_inject/2]
     32       2 S [migration/2]
     33       2 S [ksoftirqd/2]
     35       2 I [kworker/2:0H-events_highpri]
```

(d) List the pid, ppid, and name of the command of process with pid 123 (or any PID).

ANS →

Command → ps -o pid,ppid,comm -p PID

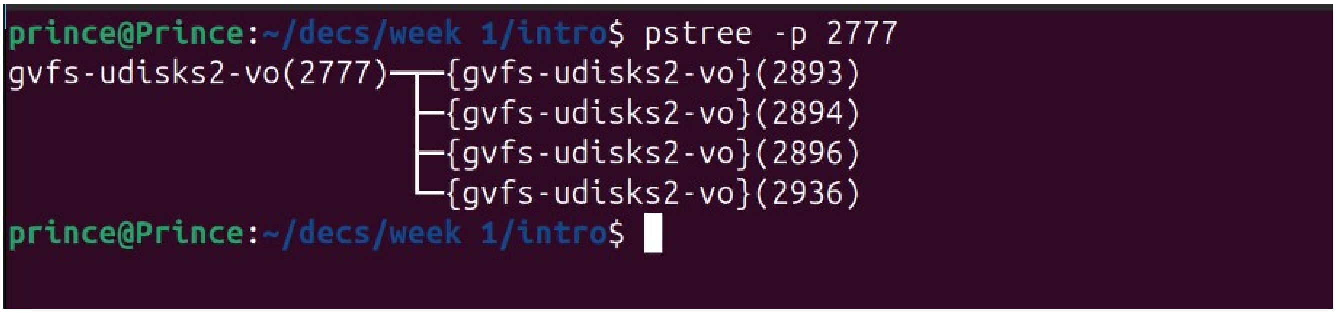


```
prince@Prince:~/decs/week 1/intro$ ps -o pid,ppid,comm -p 5859
    PID    PPID COMMAND
   5859    5852 bash
prince@Prince:~/decs/week 1/intro$ 
```

(e) Print the process tree of a process with pid 123 using ascii characters.

ANS →

Command→ pstree -p PID



```
prince@Prince:~/decs/week 1/intro$ pstree -p 2777
gvfs-udisks2-vo(2777)─┬─{gvfs-udisks2-vo}(2893)
                      ├─{gvfs-udisks2-vo}(2894)
                      ├─{gvfs-udisks2-vo}(2896)
                      └─{gvfs-udisks2-vo}(2936)
prince@Prince:~/decs/week 1/intro$ 
```

2. Answer the following questions by looking at the files in the proc filesystem, and the outputs of the lscpu, uname, uptime commands on your system.

(a) How many CPUs / cores / processors does your machine have? What is the frequency of each processor? What is the architecture of your CPU?

ANS →

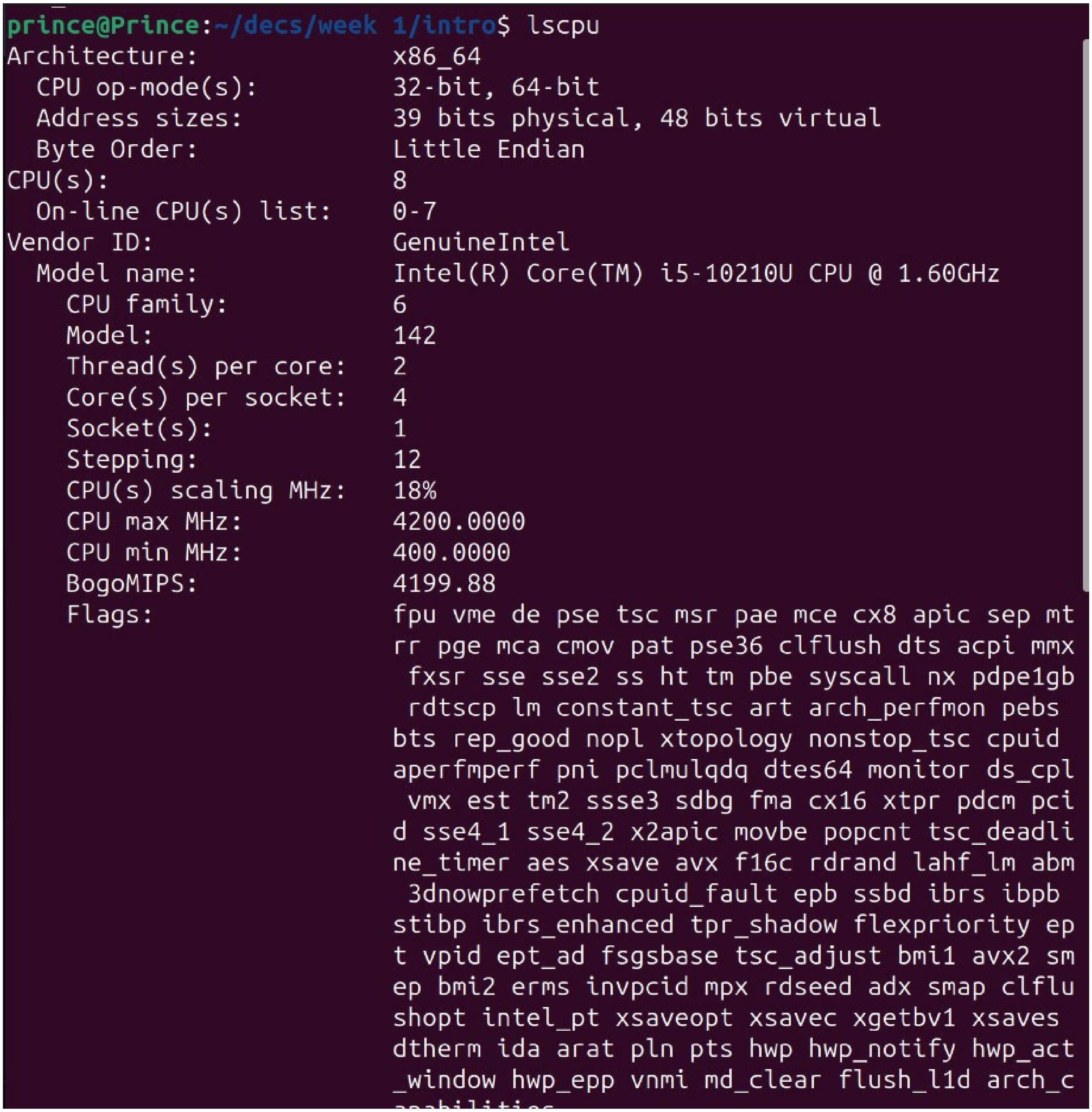
Command → lscpu

Cores = 8

Frequency →

```
CPU max MHz:         4200.0000
CPU min MHz:         400.0000
```

Architecture = x86_64

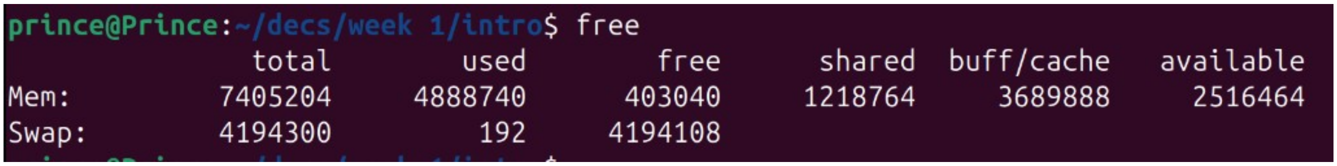


```
prince@Prince:~/decs/week 1/intro$ lscpu
Architecture:             x86_64
  CPU op-mode(s):         32-bit, 64-bit
  Address sizes:          39 bits physical, 48 bits virtual
  Byte Order:             Little Endian
CPU(s):                   8
  On-line CPU(s) list:    0-7
Vendor ID:                GenuineIntel
  Model name:             Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz
    CPU family:           6
    Model:                142
    Thread(s) per core:   2
    Core(s) per socket:   4
    Socket(s):            1
    Stepping:             12
    CPU(s) scaling MHz:   18%
    CPU max MHz:          4200.0000
    CPU min MHz:          400.0000
    BogoMIPS:             4199.88
    Flags:                fpu vme de pse tsc msr pae mce cx8 apic sep mt
                          rr pge mca cmov pat pse36 clflush dts acpi mmx
                           fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb
                           rdtscp lm constant_tsc art arch_perfmon pebs
                          bts rep_good nopl xtopology nonstop_tsc cpuid
                          aperfmperf pni pclmulqdq dtes64 monitor ds_cpl
                           vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pci
                          d sse4_1 sse4_2 x2apic movbe popcnt tsc_deadli
                          ne_timer aes xsave avx f16c rdrand lahf_lm abm
                           3dnowprefetch cpuid_fault epb ssbd ibrs ibpb
                          stibp ibrs_enhanced tpr_shadow flexpriority ep
                          t vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 sm
                          ep bmi2 erms invpcid mpx rdseed adx smap clflu
                          shopt intel_pt xsaveopt xsavec xgetbv1 xsaves
                          dtherm ida arat pln pts hwp hwp_notify hwp_act
                          _window hwp_epp vnmi md_clear flush_l1d arch_c
                          apabilities
```

(b) How much physical memory does your system have? How much of this memory is free?

ANS →

Command → free
Total memory = 7405204
Free memory = 403040

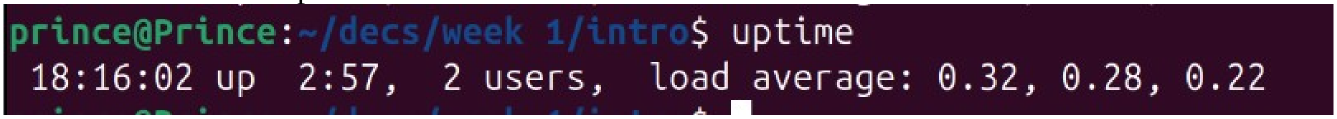


```
prince@Prince:~/decs/week 1/intro$ free
               total        used        free      shared  buff/cache   available
Mem:         7405204     4888740      403040     1218764     3689888     2516464
Swap:        4194300         192     4194108
```

(c) For how long has your system been running? What is total number of context switches since the system booted up?

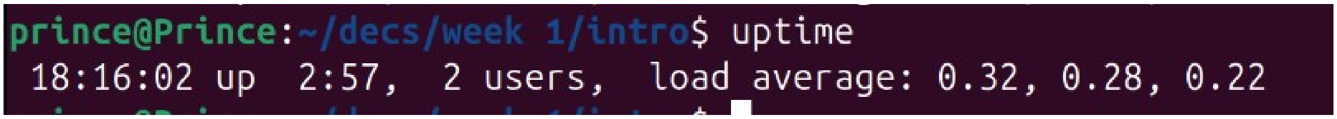
ANS →

Command → uptime



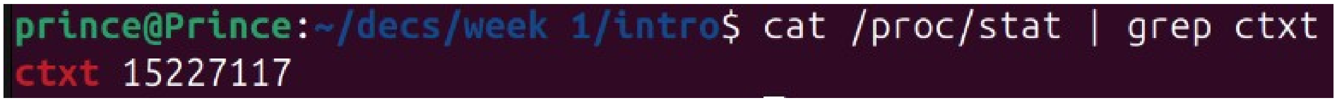
```
prince@Prince:~/decs/week 1/intro$ uptime
 18:16:02 up  2:57,  2 users,  load average: 0.32, 0.28, 0.22
```

Command → uptime -p
Up time = 2 hours, 58 minutes



```
prince@Prince:~/decs/week 1/intro$ uptime
 18:16:02 up  2:57,  2 users,  load average: 0.32, 0.28, 0.22
```

Command → cat /proc/stat | grep ctxt
Context switches = 15227117

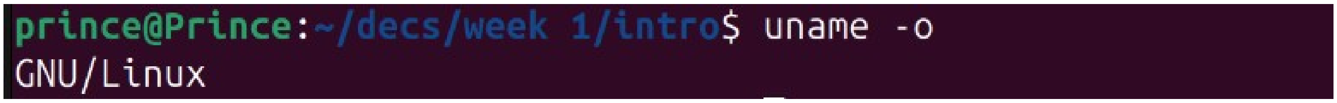


```
prince@Prince:~/decs/week 1/intro$ cat /proc/stat | grep ctxt
ctxt 15227117
```

(d) What is name of your operating system? What is the kernel version?

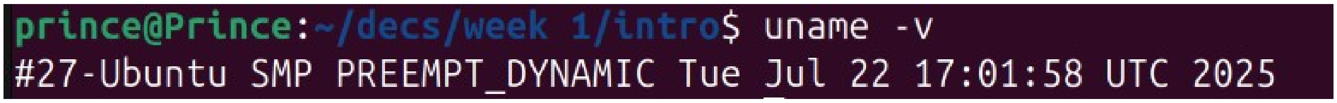
ANS →

Command → uname -o
Operating System = GNU/Linux



```
prince@Prince:~/decs/week 1/intro$ uname -o
GNU/Linux
```

Command → uname -v
Kernel version = #27-Ubuntu SMP PREEMPT_DYNAMIC Tue Jul 22 17:01:58 UTC 2025



```
prince@Prince:~/decs/week 1/intro$ uname -v
#27-Ubuntu SMP PREEMPT_DYNAMIC Tue Jul 22 17:01:58 UTC 2025
```

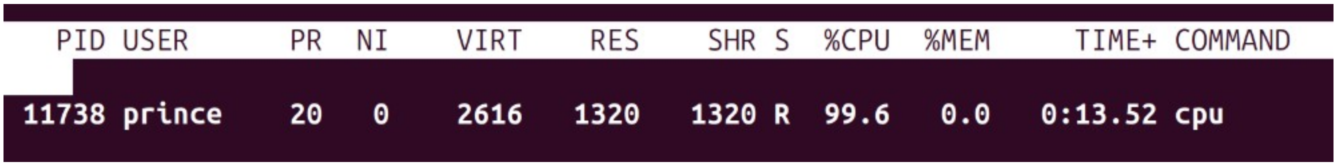
3. In this question, we will understand how to monitor the status of a running process using the top command. Compile the program cpu.c given to you and execute it in the shell. $ gcc cpu.c -o cpu $ ./cpu This program runs in an infinite loop without terminating. Now open another terminal, run the top command and answer the following questions about the cpu process.

(a) What is the PID of the process running the cpu command?

ANS →

Command → top

PID = 11738



```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND

11738 prince    20   0    2616   1320   1320 R  99.6   0.0   0:13.52 cpu
```

(b) How much CPU and memory does this process consume?

ANS →

CPU = 99.9%

Memory = 0%

(c) What is the current state of the process? For example, is it running or in a blocked state or a zombie state?

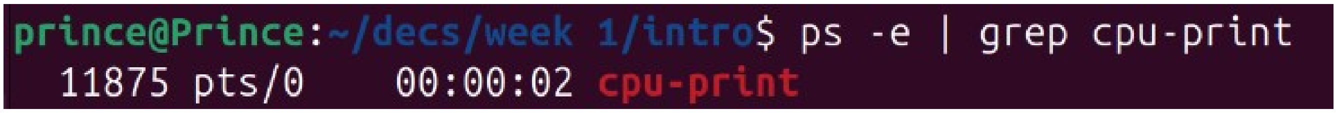
ANS →

State = Running

4. In this question, we will understand how the Linux shell (e.g., the bash shell) runs user commands by spawning new child processes to execute the various commands.

(a) Compile the program cpu-print.c given to you and execute it in the bash or any other shell of your choice as follows. $ gcc cpu-print.c -o cpu-print $ ./cpu-print
This program runs in an infinite loop printing output to the screen. Now, open another terminal and use the ps command with suitable options to find out the pid of the process spawned by the shell to run the cpu-print executable.

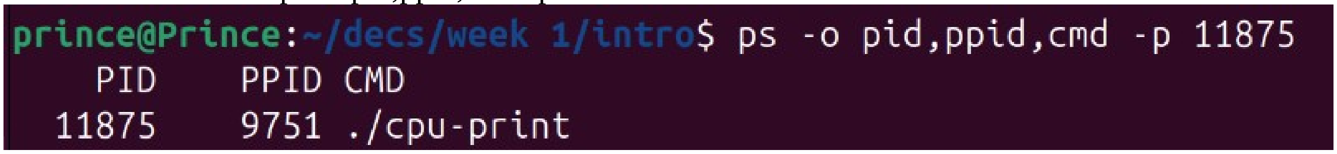
ANS →
Command → ps -e | grep cpu-print
PID = 11875



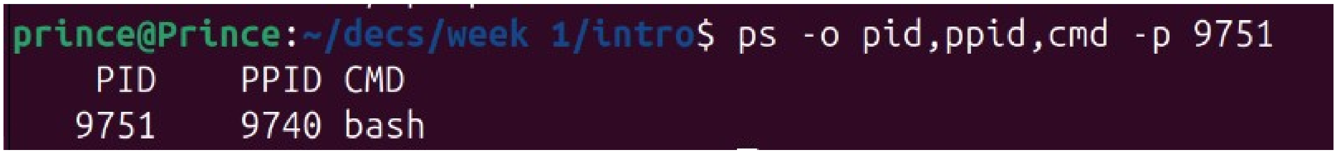
```
prince@Prince:~/decs/week 1/intro$ ps -e | grep cpu-print
  11875 pts/0    00:00:02 cpu-print
```

(b) Find the PID of the parent of the cpu-print process, i.e., the shell process. Next, find the PIDs of all the ancestors, going back at least 5 generations (or until you reach the init process).

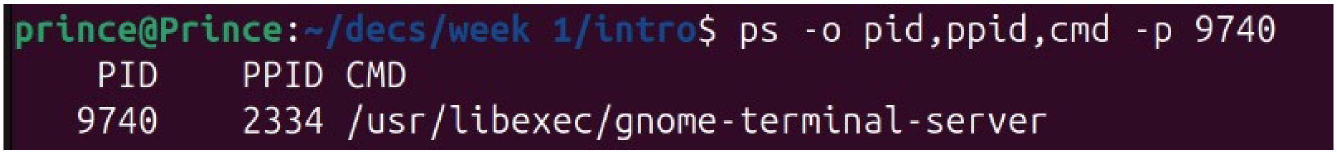
ANS →
Command → ps -o pid,ppid,cmd -p PID



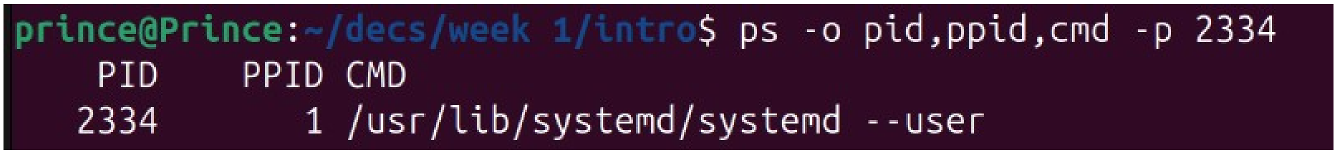
```
prince@Prince:~/decs/week 1/intro$ ps -o pid,ppid,cmd -p 11875
    PID    PPID CMD
  11875    9751 ./cpu-print
```



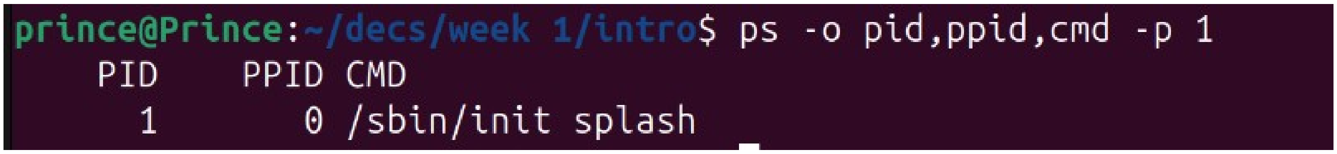
```
prince@Prince:~/decs/week 1/intro$ ps -o pid,ppid,cmd -p 9751
    PID    PPID CMD
   9751    9740 bash
```



```
prince@Prince:~/decs/week 1/intro$ ps -o pid,ppid,cmd -p 9740
    PID    PPID CMD
   9740    2334 /usr/libexec/gnome-terminal-server
```



```
prince@Prince:~/decs/week 1/intro$ ps -o pid,ppid,cmd -p 2334
    PID    PPID CMD
   2334       1 /usr/lib/systemd/systemd --user
```

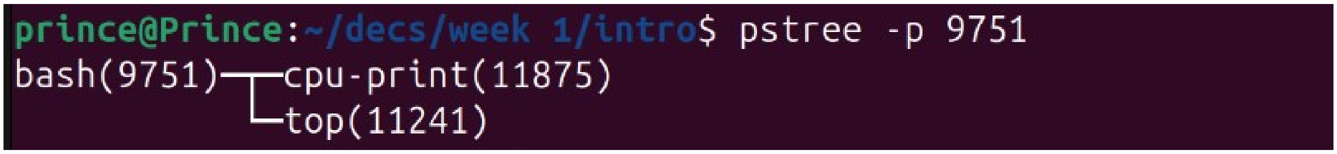


```
prince@Prince:~/decs/week 1/intro$ ps -o pid,ppid,cmd -p 1
    PID    PPID CMD
      1       0 /sbin/init splash
```

(c) Using the pstree command, print the process tree of the shell process that is running your cpu-print program.

ANS →

Command → pstree -p PID

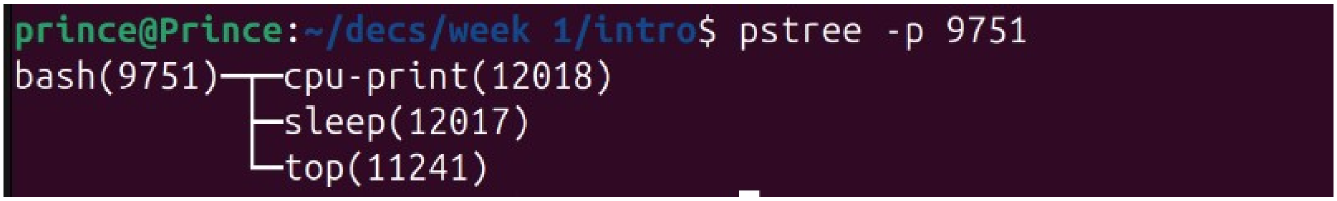


```
prince@Prince:~/decs/week 1/intro$ pstree -p 9751
bash(9751)─┬─cpu-print(11875)
           └─top(11241)
```

(d) Now, stop the execution of the cpu-print executable. Run a long background command, e.g., sleep 10000 & on your shell. Restart the long cpu-print process in the foreground again. Visualize the process tree of your shell again.

ANS →

Commands → pstree -p PID



```
prince@Prince:~/decs/week 1/intro$ pstree -p 9751
bash(9751)─┬─cpu-print(12018)
           ├─sleep(12017)
           └─top(11241)
```

5. When you type in a command into the shell, the shell does one of two things. For some commands, executables that perform that functionality already come with your Linux kernel installation. For such commands, the shell simply invokes the executable to run the command. For other commands where the executable does not exist, the shell implements the command itself within its code. Consider the following commands that you can type in the bash shell: cd, ls, ps, sleep, history. Which of these commands already exist as executables in the Linux kernel directory tree, and which are implemented by the bash code itself? If the executable already exists, what is the pathname of the executable file? You may use the which command to help you.

ANS →

Commands in kernel = cd , history

Commands implemented in bash code →

- ls = /usr/bin/ls
- ps = /usr/bin/ps
- sleep = /usr/bin/sleep