

# CSE3241: Operating System and System Programming

Class-4

Sangeeta Biswas, Ph.D.

Assistant Professor

Dept. of Computer Science and Engineering (CSE)

Faculty of Engineering

University of Rajshahi (RU)

Rajshahi-6205, Bangladesh

E-mail: [sangeeta.cse@ru.ac.bd](mailto:sangeeta.cse@ru.ac.bd)

# Multitasking OS

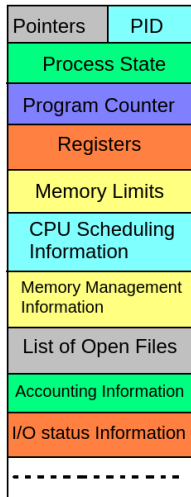
Multi-tasking OS allows a user to perform multiple computer tasks using a single set of resources in such a way that user get pseudo-parallel feeling, i.e., the user feels all of her/his tasks are running in parallel.

- ▶ Almost all modern OS, such as Microsoft Windows 2000, IBM's OS/390, and Linux, have multi-tasking capability.
- ▶ The main aim of multitasking is to ensure maximum utilization of advanced CPU, which is much more faster than the old time CPU, by keeping it busy at the maximum time.
- ▶ In multitasking, OS switches execution power from one process to another frequently so that each process has a progress instead of waiting until a specific process completely finished.
- ▶ OS needs to keep track of all processes to do multitasking smoothly, therefore, it uses **Process Control Block**.

## Process Control Block

Process Control Block (PCB) is a block of information that represent a process in an OS. It is also known as a **task control block**.

- ▶ **Process state:** running, waiting, etc.
- ▶ **Program counter:** location of instruction to next execute.
- ▶ **CPU registers:** contents of all process-centric registers
- ▶ **CPU scheduling information:** priorities, scheduling queue pointers.
- ▶ **Memory limits:** memory allocated to the process.
- ▶ **Accounting information:** CPU used, clock time elapsed since start, time limits.
- ▶ **I/O status information:** I/O devices allocated to process.



# PID

PID or Process number is a unique number of a process assigned by the OS so that it can easily keep track all current processes.

- ▶ It is only valid until the process is properly terminated in the current session.
  - ▶ If ,for some reasons, a process is killed/ finished execution, but do not get a chance to properly inform its parent process, then it holds the process ID.
- ▶ For a new start of a program, it gets a new PID.
  - ▶ So, if we run our code at different time/ different sessions, our program will become a new process and get a different PID.
- ▶ As long as a process has its PID, it has an entry in the **Process Table**.

# Investigate PID and PCB in Ubuntu

Commands to see PID and PCBs of all processes, currently managed by the OS including its own processes:

- For PID:
    - \$ top
  - For PCB:
    - \$ ls /proc
    - \$ ls /proc/<PID>/
- e.g., \$ ls /proc/3668

```
Activities Terminal
File Edit View Search Terminal Tabs Help
cse@cse-M5-7809:~$ top
top - 05:48:16 up 1:17, 1 user, load average: 1.18, 1.15, 1.11
Tasks: 539 total, 2 running, 365 sleeping, 0 stopped, 0 zombie
NCPU(s): 3.0 us, 0.1 sy, 0.0 ni, 96.9 id, 0.0 wa, 0.0 hi, 0.0 st, 0.0 st
KlB Mem : 32815428 total, 15331540 free, 11359360 used, 6124528 buff/cache
KlB Swap: 2097148 total, 2097148 free, 0 used, 20748616 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR  S  NCPUP NMEN  TIME+ COMMAND
 3660 cse        20   0 30.460g 7.004g 665200 R 96.0 24.9 59:20.53 python
2119 cse        20   0 1044508 64120 49524 S 0.7 0.2 0:02.61 skypeforlinux
1214 root      -51   0 0 0 0 S 0.3 0.0 0:27.01 lrq/124-nvidia
1794 cse        20   0 5879636 341796 105044 S 0.3 1.0 1:56.09 gnome-shell
2148 cse        20   0 5791800 253180 119012 S 0.3 0.8 0:12.49 skypeforlinux
2407 cse        20   0 3793640 415344 204932 S 0.3 1.3 4:00.78 firefox
2490 cse        20   0 3161092 439084 144036 S 0.3 1.3 0:37.78 Web Content
3409 cse        20   0 794408 38516 28032 S 0.3 0.1 0:01.73 gnome-terminal-
4120 cse        20   0 2950068 268624 136912 S 0.3 0.8 1:44.69 Web Content
6547 cse        20   0 3104956 376556 129352 S 0.3 1.1 0:08.90 Web Content
7536 cse        20   0 44936 4692 3528 R 0.3 0.0 0:00.06 top
  1 root      20   0 225584 9276 6688 S 0.0 0.0 0:05.00 systemd
  2 root      20   0 0 0 0 S 0.0 0.0 0:00.01 kthreadd
  3 root      0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_gp
  4 root      0 -20 0 0 0 I 0.0 0.0 0:00.00 rcu_par_gp
  6 root      0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/0:0H-kb
  8 root      20   0 0 0 0 I 0.0 0.0 0:00.18 kworker/u256:0-
10 root      0 -20 0 0 0 I 0.0 0.0 0:00.00 nm_percpu_wq
11 root      20   0 0 0 0 S 0.0 0.0 0:00.01 ksoftirqd/0
12 root      20   0 0 0 0 I 0.0 0.0 0:00.73 rcu_sched
13 root      rt  0 0 0 0 S 0.0 0.0 0:00.00 migration/0
14 root     -51   0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/0
15 root      20   0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/0
16 root      20   0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/1
17 root     -51   0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/1
18 root      rt  0 0 0 0 S 0.0 0.0 0:00.13 migration/1
19 root      20   0 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/1
21 root      0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/1:0H-kb
22 root      20   0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/2
23 root     -51   0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2
24 root      rt  0 0 0 0 S 0.0 0.0 0:00.13 migration/2
25 root      20   0 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/2
27 root      0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/2:0H-kb
28 root      20   0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/3
29 root     -51   0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/3
30 root      rt  0 0 0 0 S 0.0 0.0 0:00.13 migration/3
31 root      20   0 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/3
33 root      0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/3:0H-kb
34 root      20   0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/4
35 root     -51   0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/4
36 root      rt  0 0 0 0 S 0.0 0.0 0:00.14 migration/4
37 root      20   0 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/4
39 root      0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/4:0H-kb
40 root      20   0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/5
41 root     -51   0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/5
42 root      rt  0 0 0 0 S 0.0 0.0 0:00.14 migration/5
```

# Process Table

The process table is an array of PCBs.

- ▶ It logically contains a PCB for each of the current process in the system.
- ▶ OS maintains pointers to each process's PCB in the process table so that it can access the PCB quickly.

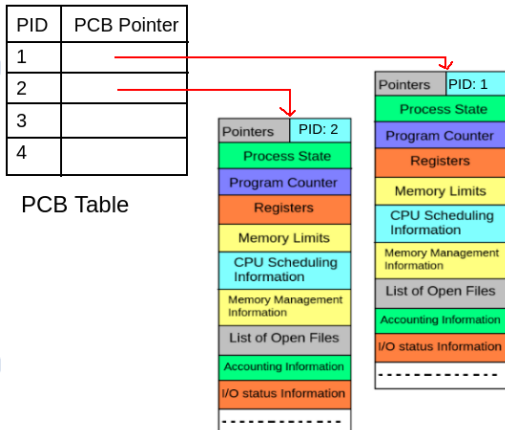


Figure: PCB table

# Context Switch

A context switch occurs when the CPU switches from one process to another.

- ▶ OS saves the state of the old process and load the saved state for the new process via a context switch.

