Process_Management.md

Process Management

A process is an instance (i.e., running) of a program. Or simply a program in execution is called as a process. While a package/application/program is an executable file lying in some directory in the hard drive (storage), a process is started whent he user or another program calls (initializes) the program and this is opened in RAM and CPU allocates time to this.

There are fundamentally two types of processes in Linux:

- Foreground processes:
  Also known as interactive processes. Foreground processes are initialized and controlled through a terminal session.
- Background processes:
  Also known as non-interactive/automatic processes. Background processes are not connected to a terminal; they don’t expect any user input.

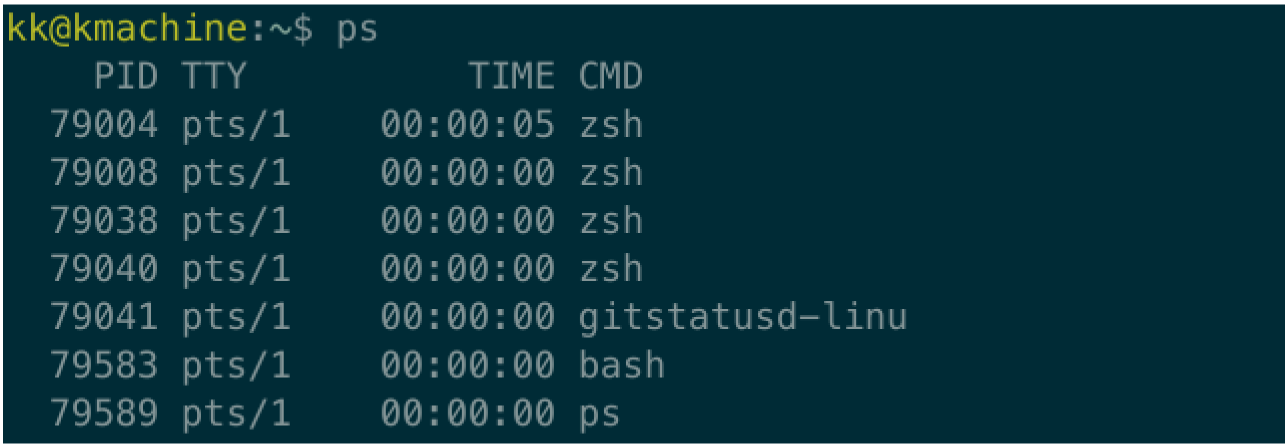
Every process has a parent. The top most process is `init`, whose process id is 1

Stages of a process:

1. User-running
2. Kernel-running
3. Ready to run in memory
4. Asleep in Memory
5. Ready to run, swapped
6. Sleep, Swapped
7. Pre-empted
8. Created
9. Zombie.

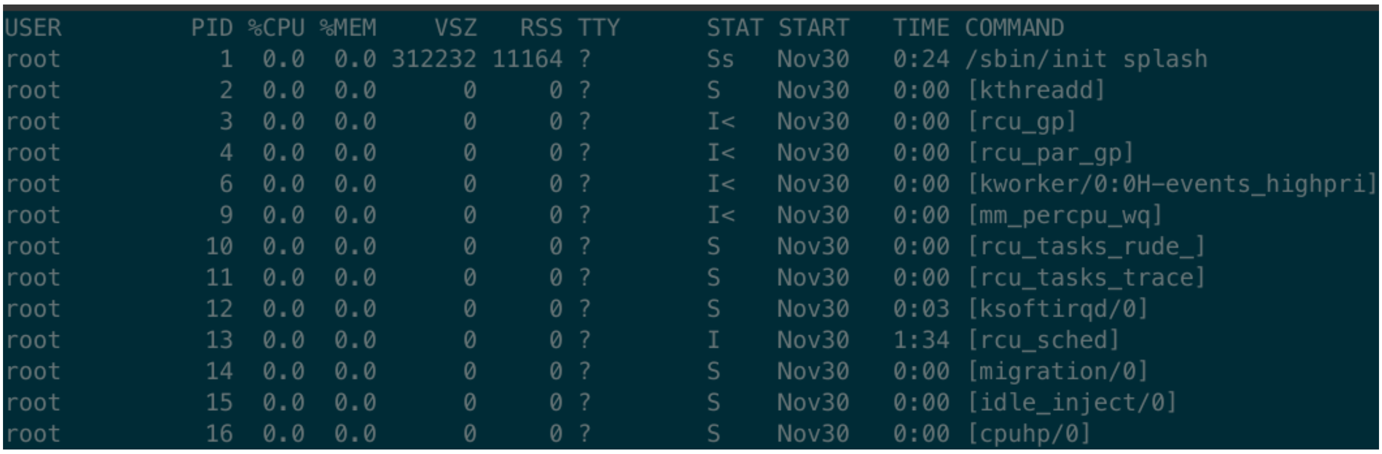
ps

`ps` is the command to check all the processes and their resource utilization (CPU & memory)



```
kk@kmachine:~$ ps
    PID TTY          TIME CMD
  79004 pts/1    00:00:05 zsh
  79008 pts/1    00:00:00 zsh
  79038 pts/1    00:00:00 zsh
  79040 pts/1    00:00:00 zsh
  79041 pts/1    00:00:00 gitstatusd-linu
  79583 pts/1    00:00:00 bash
  79589 pts/1    00:00:00 ps
```

`ps aux` gives processes started by all the users on the Computer



```
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0 312232 11164 ?        Ss   Nov30   0:24 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    Nov30   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   Nov30   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   Nov30   0:00 [rcu_par_gp]
root           6  0.0  0.0      0     0 ?        I<   Nov30   0:00 [kworker/0:0H-events_highpri]
root           9  0.0  0.0      0     0 ?        I<   Nov30   0:00 [mm_percpu_wq]
root          10  0.0  0.0      0     0 ?        S    Nov30   0:00 [rcu_tasks_rude_]
root          11  0.0  0.0      0     0 ?        S    Nov30   0:00 [rcu_tasks_trace]
root          12  0.0  0.0      0     0 ?        S    Nov30   0:03 [ksoftirqd/0]
root          13  0.0  0.0      0     0 ?        I    Nov30   1:34 [rcu_sched]
root          14  0.0  0.0      0     0 ?        S    Nov30   0:00 [migration/0]
root          15  0.0  0.0      0     0 ?        S    Nov30   0:00 [idle_inject/0]
root          16  0.0  0.0      0     0 ?        S    Nov30   0:00 [cpuhp/0]
```

PID - Process ID

TTY - Controlling Terminal

STAT: Process State Code

TIME: Total time of CPU Usage

CMD: The command that triggered the process

RSS: Both swap memory and physical Memroy

VSZ: Virtual memory usage of the process

%CPU: CPU Time used by the process run time

%MEM: Processes set size to the physical memory on the machine

START: Process Start time

Various Process States:

D - Uninterruptible sleep

I - Idle kernel thread

R - Running or runnable (in the queue)

S - Interruptible sleep (waiting for event or input)

T - Stopped by job control signal

t - Stopped by debugger

X - dead (generally, not seen)

Z - Zombie

Kill

`kill` is the command to stop any process
`kill <pid>`

To force kill a program :

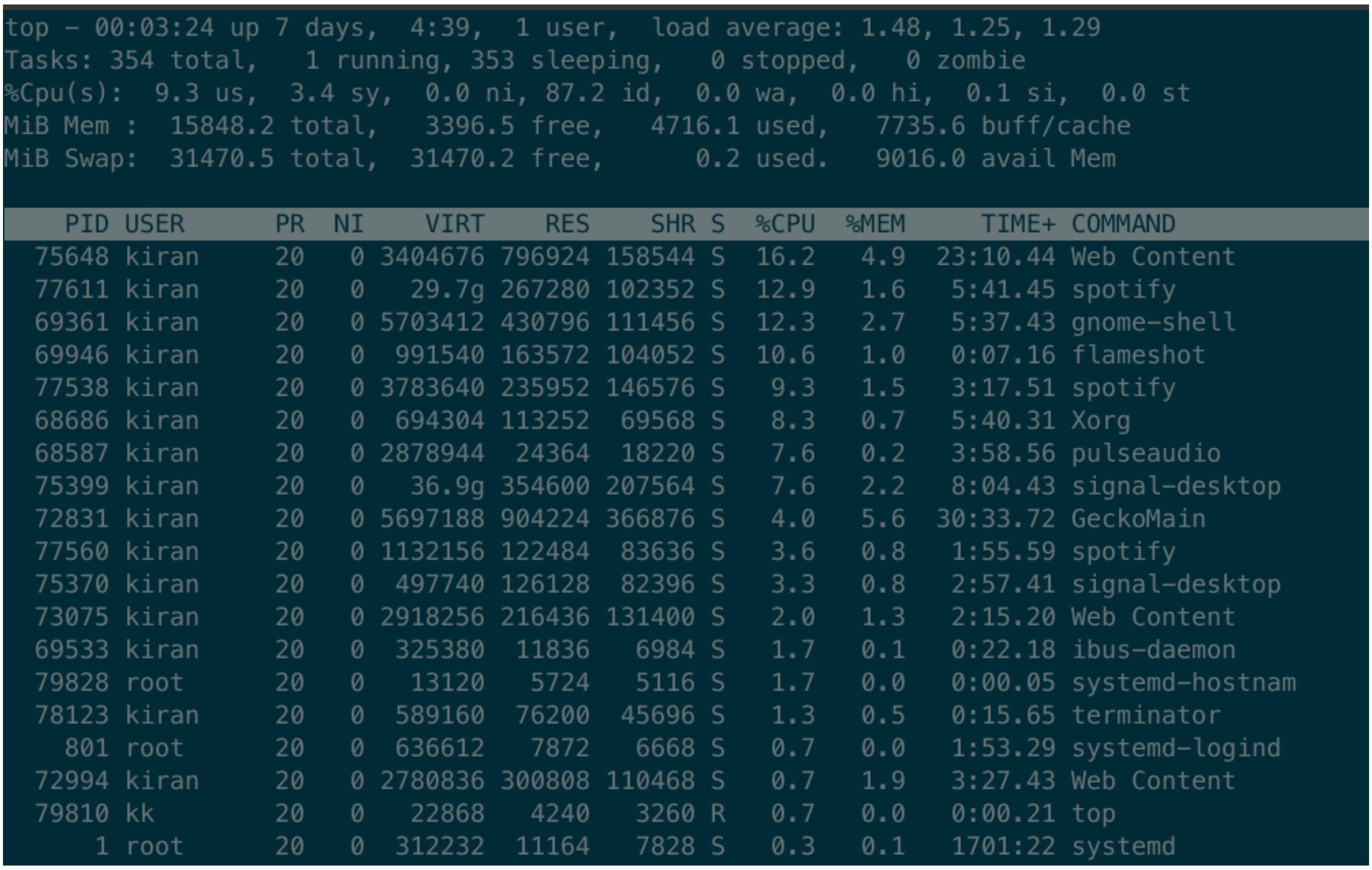
`kill -9 <pid>`

Top

This command gives real time information about the processes adn their utilization

The data get refreshed every 3 seconds. You could change it by pressing 's'

The processes are sorted by CPU utilization by default. IF you want to change that you could press 'f' and select other field.



```
top - 00:03:24 up 7 days,  4:39,  1 user,  load average: 1.48, 1.25, 1.29
Tasks: 354 total,   1 running, 353 sleeping,   0 stopped,   0 zombie
%Cpu(s):  9.3 us,  3.4 sy,  0.0 ni, 87.2 id,  0.0 wa,  0.0 hi,  0.1 si,  0.0 st
MiB Mem :  15848.2 total,   3396.5 free,   4716.1 used,   7735.6 buff/cache
MiB Swap:  31470.5 total,  31470.2 free,      0.2 used.   9016.0 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  75648 kiran     20   0 3404676 796924 158544 S  16.2   4.9  23:10.44 Web Content
  77611 kiran     20   0   29.7g 267280 102352 S  12.9   1.6   5:41.45 spotify
  69361 kiran     20   0 5703412 430796 111456 S  12.3   2.7   5:37.43 gnome-shell
  69946 kiran     20   0  991540 163572 104052 S  10.6   1.0   0:07.16 flameshot
  77538 kiran     20   0 3783640 235952 146576 S   9.3   1.5   3:17.51 spotify
  68686 kiran     20   0  694304 113252  69568 S   8.3   0.7   5:40.31 Xorg
  68587 kiran     20   0 2878944  24364  18220 S   7.6   0.2   3:58.56 pulseaudio
  75399 kiran     20   0   36.9g 354600 207564 S   7.6   2.2   8:04.43 signal-desktop
  72831 kiran     20   0 5697188 904224 366876 S   4.0   5.6  30:33.72 GeckoMain
  77560 kiran     20   0 1132156 122484  83636 S   3.6   0.8   1:55.59 spotify
  75370 kiran     20   0  497740 126128  82396 S   3.3   0.8   2:57.41 signal-desktop
  73075 kiran     20   0 2918256 216436 131400 S   2.0   1.3   2:15.20 Web Content
  69533 kiran     20   0  325380  11836   6984 S   1.7   0.1   0:22.18 ibus-daemon
  79828 root      20   0   13120   5724   5116 S   1.7   0.0   0:00.05 systemd-hostnam
  78123 kiran     20   0  589160  76200  45696 S   1.3   0.5   0:15.65 terminator
    801 root      20   0  636612   7872   6668 S   0.7   0.0   1:53.29 systemd-logind
  72994 kiran     20   0 2780836 300808 110468 S   0.7   1.9   3:27.43 Web Content
  79810 kk        20   0   22868   4240   3260 R   0.7   0.0   0:00.21 top
      1 root      20   0  312232  11164   7828 S   0.3   0.1   1701:22 systemd
```

Shift + p -- Sorts the processes by highest CPU utilization
Shift + m -- Sorts the processes by highest Memory utilization

Fields in the Header

```
us: Amount of time the CPU spends executing processes for people in 'user space'
sy: Amount of time spent running system's kernel space's processes.
```

```
ni: Amount of time spent executing processes with a manually set nice value.
id: Amount of CPU idle time.
wa: Amount of time the CPU spends waiting for I/O to complete.
hi: Amount of time spent servicing hardware interrupts.
si: Amount of time spent servicing software interrupts.
st: Amount of time lost due to running virtual machines ('steal time')
```

Fields in main output

```
PID: Shows task's unique process id.
USER: User name of owner of task
PR: Stands for priority of the task.
NI: Represents a Nice Value of task. A Negative nice value implies higher priority, and positive Nice value means lower priorityUSER: User name of owner of task.
SHR: Represents the amount of shared memory used by a task.
VIRT: Total virtual memory used by the task.
%CPU: Represents the CPU usage.
TIME+: CPU Time, the same as TIME, but reflecting more granularity through hundredths of a second.
%MEM: Shows the Memory usage of task.
```

Sorting

By default top sorts the entries by CPU usage.
Press `M` (upper case) to sort by Memory
To revert to sortby CPU usage press `P` (upper case)

Priority (PR) & Niceness (NI)

Priority and Niceness are related. While Priority is

PR is the priority level. The lower the PR, the higher the priority of the process will be.
NI is niceness of the process. The higher the niceness, the process will get lower precedence.

PR value can be computed by the following formula: PR = 20 + NI.
the process with niceness 3 has the priority 23 (20 + 3) and the process with niceness -7 has the priority 13 (20 - 7). You can check the first by running command nice -n 3 top. It will show that top process has NI 3 and PR 23. But for running nice -n -7 top in most Linux systems you need to have root privileges because actually the lower PR value is the higher actual priority is. Thus the process with PR 13 has higher priority than processes with standard priority PR 20.

Niceness ranges -20 to 20. Lesser the Niceness higher the priority.
nice and renice commands are used to update Priority of the user processes. PR of only user processes can be altered with nice and renice commands

nice: A program can be started with a specific niceness with the below command

```
nice -n <nice_value> ./myProgram
```

Example

```
nice -n 9 ./myscript.sh
```

renice: Priority of a program can be altered with renice command

```
renice -n <nice_value> <PID>
```

Example

```
renice -n 9 1344
```

load average :

Load Average is the measure of the load on the processors. Load Average has three fields each of which is the number of processes waiting for CPU in the last 1 minute, last 5 minutes, and last 15 minutes.

For a Computer with 1 CPU, the load average should always be less than 1 and above 0.9 indicates high utilization of CPU. For a computer with 2 CPUs, the uptime should be under 2 and so on so forth

Further Reading

Understanding ps command
A guide to Linux TOP command