Contents

[**TOP Command:** 2](#_Toc495656843)

[**CPU Usage:** 3](#_Toc495656844)

[**Load:** 3](#_Toc495656845)

[**Load average:** 3](#_Toc495656846)

[ Load average: traffic analogy 4](#_Toc495656847)

[**Sleeping Process:** 5](#_Toc495656848)

[**Zombie process:** 5](#_Toc495656849)

[ How Process dies: 5](#_Toc495656850)

[ Getting rid of zombie processes: 6](#_Toc495656851)

[**Nice Value:** 6](#_Toc495656852)

[**Priority:** 7](#_Toc495656853)

[**Steal time:** 7](#_Toc495656854)

[ What if steal time is well above zero? 7](#_Toc495656855)

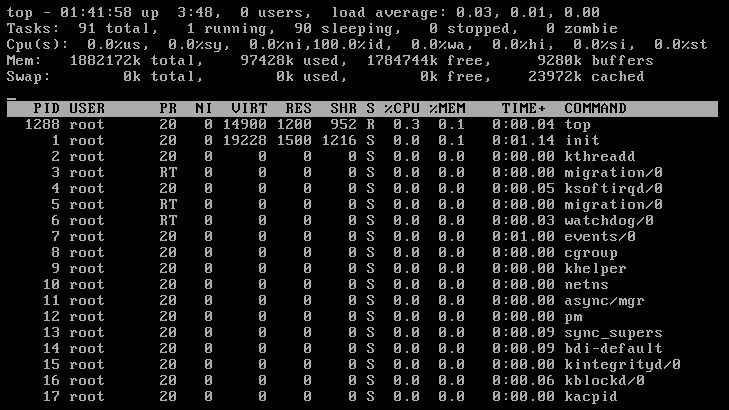
[**Explanation of output in table form:** 8](#_Toc495656856)

[**Shortcuts** 9](#_Toc495656857)

[**Notes:** 9](#_Toc495656858)

# **TOP Command:**

* Top command is used to show all running processes, CPU usage of particular process, memory usage of particular process, etc.
* Command: top



* In 1st line of output: time, users, load average

Current time- 01:41:58

Uptime of the system- 3:48 hours (up)

Number of users sessions logged in -0 users

Load average- 0.03, 0.01, 0.00

* In 2nd line of output: tasks/processes

Total number of tasks/processes- 91 (total)

Number of running tasks/processes- 1 (running)

Number of sleeping tasks/processes- 90 (sleeping))

Number of stopped tasks- 0 (stopped)

Number of zombie tasks- 0 (zombie)

* In 3rd line of output: CPU usage

CPU usage as a percentage by user processes- 0.0% (us)

CPU usage as a percentage by system processes- 0.0% (sy)

CPU usage as a percentage by processes with priority upgrade nice- 0.0%(ni)

CPU usage as a percentage by idle processes- 0.0% (id)

CPU usage as a percentage by processes waiting for I/O operations- 0.0% (wa)

CPU usage as a percentage by processes serving hardware interrupts- 0.0% (hi)

CPU usage as a percentage by processes serving software interrupts - 0.0% (si)

CPU usage as a percentage by processes of steal time- 0.0% (st)

## **CPU Usage:**

* CPU usage is amount of usage of the processor.
* CPU usage by system (sy): Amount of CPU used for running kernel code (device drivers and kernel modules).
* CPU usage by User (us): Amount CPU used by user’s application.
* CPU usage (ni): CPU is used for executing certain process with modified priority
* CPU usage I/O Wait(wa): CPU waits for an I/O operation to complete
* CPU usage Idle (id): Kernel does nothing, no process is in running queue, no I/O operations going on.
* In 4th line of output: Physical memory (RAM)

Total amount RAM available- 1882172 KB (total)

Amount RAM used by processes including buffer and cached memory- 97428 KB (used)

Amount of RAM free - 1784444 KB (free)

Amount of RAM used by buffer- 9280 KB (buffers)

* In 5th line of output: SWAP memory

Total amount SWAP available- 0 KB (total)

Amount of SWAP memory used- 0 KB (used)

Amount SWAP memory free- 0 KB (free)

Amount of cache memory- 23972 KB (cached)

## **Load:**

* Load is a measure of the amount of computational work that a computer system performs.

## **Load average:**

* Load average is average load over a period of time.
* Separate load number doesn’t make sense. Load varies minute to minute. So load over certain period of time is taken into consideration.
* Load average is displayed by two commands: top command & uptime command.
* load average: 0.03, 0.01, 0.00
* These three numbers represent load averages over progressively 1 minute, 5 minutes and 15 minutes respectively.

### Load average: traffic analogy

* A single core CPU is like single lane of traffic.
* Many cars run on bridge. To estimate traffic on bridge, certain numbers (load average) are used.
* 0.00 means no traffic on the bridge. No car is running/ waiting.
* 1.00 means bridge is exactly at its capacity. All cars are running smoothly. No car is on waiting mode. Although if traffic gets a little heavier, it will create problem.
* More than 1.00 means some cars are in waiting mode. For an example, 2.00 means there are 2 lanes worth of cars total, one lane is worth on the bridge and one lane is worth waiting.
* So cars are just like processes and length of lane is run queue length.
* Load average 0.00 means no process is running. Idle computer has load average of 0.00.
* Load average 1.00 means 100% resource utilization.
* Load average value should be as low as possible.
* Ideal value for load average is below 1.00.
* Generally, system administrator set load average value of 0.70. It means if load average stays more than 0.70, it’s time to investigate before things getting worse.
* If load value is above 1.00, it means there is a problem.
* For an example, load average 1.05 means computer was overloaded by 5%, 5 processes were waiting to get CPU.
* For an example, load average 5.4 means computer was overloaded by 404%, 4.04 processes were waiting to get CPU.
* For an example, load average 0.70 means CPU was idled for 30% of time.
* Above explanation is applicable to single core single processor.
* For multicore processors, number of cores = maximum load no matter how cores are spread over CPUs. Load average 2.00 for dual core processor means 100% utilization.
* Load average over 15 minutes should be observed because load average over 1 minutes vary often and doesn’t give any conclusion.
* To know number of cores or processors,

Command: cat /proc/cpuinfo | grep “model name” | wc -l



## **Sleeping Process:**

* When a process starts, it gets loaded into memory so some memory has to be allocated to that process. Moreover, it also gets some processor time, additional memory to hold run data, OS resources. So process request for these resources. It might be possible that kernel is not able to fulfill all requirement immediately, perhaps requested resources are busy or other processes are accessing resources. In such a case, when process starts and requests for resources, that process will be put into sleep mode until resource gets free.
* So sleeping process is a process that is waiting for getting requested resources.
* Sleeping process is identified by S in S (state) column of output of top command.
* Moreover, processes can be put on waiting mode manually as well by sleep command.
* Command: sleep <count>

Example: sleep 30: It causes current terminal session to wait for 30 seconds.

Example: sleep 3600; mplayer v.mp3: It causes mplayer process to wait for 3600 seconds (1 hour).

## **Zombie process:**

* Zombie means dead.
* Zombie process is a process that is already dead but not removed completely from memory.

### How Process dies:

* When a process dies in Linux, it isn’t removed from memory immediately.
* Its process descriptor stays in memory (process descriptor takes only small amount of memory).
* Now process status becomes EXIT\_ZOMBIE.
* Its parent process is notified that its child process has died with SIGCHLD signal.
* Now parent process is supposed to execute wait() system call to read dead process’s exit status and other information.
* After wait() is called by parent process, the zombie process is completely removed from memory.
* In some cases, parent process isn’t programmed properly and so never calls wait() and its zombie process (children) stick around in memory.
* Zombie process is identified by Z in S (state) column of output of top command.
* Zombie process don’t use any of system resources. Each zombie process uses a small amount of its process descriptor that is stored in memory.
* Since Linux has fix number of PIDs, other processes will not able to get PID if zombies are accumulating in a large number.

### Getting rid of zombie processes:

* Zombie process cannot be killed as they are already dead.
* To get rid of it, SIGCHLD signal is sent to its parent process. This signal tells its parent process to execute wait() system call and clean up zombie children.

Command: kill -s SIGCHLD <PID of parent process>

* However, if parent process isn’t programmed properly, it will ignore SIGCHLD signals and unable to remove zombies.
* In such a case, its parent process must be killed. When the parent process (that created zombies) ends, init inherits zombie processes and becomes new parent. Now init periodically executes wait() call and clean up zombie processes.

## **Nice Value:**

* Linux kernel schedules different processes to access required resources according to need and priority.
* When you need higher priority to some processes to get more CPU time or get resources quickly, it can be done with process niceness means process nice and renice value.
* Nice command launches a process with user defined scheduling priority.
* Renice command modify the scheduling priority of a running process.
* Range of nice value is -20 to 19.

**-20:** **highest priority**: process gets more resources, thus slowing down other processes.

**+19:** **lowest priority**: process runs slowly and has less impact on speed of other processes.

* By default, process starts with 0 nice value.
* Nice value is displayed under NI column of output of TOP command.
* To launch process with specific priority,

Command: nice -<nice value> <command/process/application> OR

Command: nice -n <nice value> <command/process/application> OR

Command: nice -<nice value> -p <PID> OR

Command: nice -n <nice value> -p <PID>

Example: nice -10 top OR nice -n 10 top:

It will set nice value +10 (positive 10) to top command.

Example: nice --10 top: OR nice -n -10 top

It will set nice value -10 (negative 10) to top command.

* Normal user (non root user) cannot launch process with higher priority.
* To change priority of running process,

Command: renice -<nice value> <command/process/application> OR

Command: renice -n <nice value> <command/process/application> OR

Command: renice -<nice value> -p <PID> OR

Command: renice -n <nice value> -p <PID>

* To change priority of all processes belongs to a group,

Command: renice -n <nice value> -g <group name>

* To change priority of all processes owned by a user,

Command: renice -n <nice value> -u <user name>

* Note:
* Above commands make changes temporary. After reboot, process will run with its default value.
* To make changes permanently, make changes in /etc/security/limits.conf file.

## **Priority:**

* Priority is process’s actual priority used by Linux kernel.
* Relation between PR (PRiority) and NI (nice value) is,

PR = 20 + NI

As NI value ranges from -20 to +19, PR value ranges from 0 to 39.

PR = 0: Highest priority

PR = 39: Lowest priority

* Linux system priority ranges from 0 to 139 in which 0 to 99 for real time and 100 to 139 for users.
* It might happen that processes having same nice value have different priorities.
* Priority value is displayed under PR column of output of TOP command.

## **Steal time:**

* Steal time is the percentage of the time a virtual CPU waits for a real CPU while the hypervisor is servicing another virtual processor.
* VM shares system resources with other VMs. VM also shares CPU cycles.
* Say, four VMs are running on one physical server. So ideally, each of them should get 25% of total CPU cycles. Theoretically, it might not possible. One VM might consume more proportion of CPU cycles whereas other VM might able to consume less CPU cycles. In such a case, VM having lower CPU cycles, might needs more CPU cycles, although as other VMs are using more CPU cycles, it is not possible to get more CPU cycles. In that case, VM needs to wait for CPU cycles until other VMs release CPU cycle. The amount of time in percentage, VM needs to wait for getting CPU resources is called steal time.

### What if steal time is well above zero?

* There are two possibilities:

1. Larger VM with more resources is needed.
2. The physical server is oversold (physical server is service more VMs than its actual capacity) and VMs are aggressively competing for resources.

* The way to determine the possibility, perform the same roles residing on different VMs, on identical two physical servers.
* If steal time is more on each VM then possibility 1. To overcome, we need VM with more CPU resources.
* If steal time is more on only one VM then physical server is oversold. To overcome, more VM to another physical server. If steal time still remains high then contact hosting provider.

## **Explanation of output in table form:**

By default, values are displayed in KB.

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Description** | **Example** |
| **PID** | Process ID of the process | 1288 |
| **USER** | Owner of the process | root |
| **PR** | Priority of the process | 20 |
| **NI** | NICE value of the process | 0 |
| **VIRT** | Virtual memory used by the process | 14900 |
| **RES** | Physical memory used by the process | 1200 |
| **SHR** | Shared memory used by the process | 952 |
| **S** | Status of the process  S- Sleep  R-Running  Z-Zombie | R |
| **%CPU** | Percentage of the CPU used by the process | 0.3 |
| **%MEM** | Percentage of RAM used the process | 0.1 |
| **TIME+** | Total time of activity of the process/ total amount of time, process is running for | 0:00.4 |
| **Command** | Name of the process | top |

|  |  |
| --- | --- |
| **Shortcuts** | |
| Key | Descriptions |
| 1 | shows individual CPUs usage |
| b | highlight running process |
| B | enable use of bold display for running processes |
| M | sort processes by memory usage in descending order |
| P | sort processes by processor (CPU) usage in descending order |
| u | sort by effective user |
| U | sort by real user |
| k | kill process |
| r | renice process |
| s | change refresh rate |
| W | save changes |
| q | quit |

## **Notes:**

* By default, top command continuously check the process states and memory details for every 3 seconds.
* In output of ps command, recent processes are at bottom end whereas in output of top command, recent processes are at top end.