**ps command in Linux with Examples**

As we all know Linux is a multitasking and multi-user systems. So, it allows multiple processes to operate simultaneously without interfering with each other. Process is one of the important fundamental concept of the Linux OS. A process is an executing instance of a program and carry out different tasks within the operating system.

Linux provides us a utility called **ps** for viewing information related with the processes on a system which stands as abbreviation for **“Process Status”.** ps command is used to list the currently running processes and their PIDs along with some other information depends on different options. It reads the process information from the virtual files in **/proc** file-system. /proc contains virtual files, this is the reason it’s referred as a virtual file system.

ps provides numerous options for manipulating the output according to our need.

**Syntax**

**ps [options]**

**Options for ps Command:**

1. **Simple process selection :** Shows the processes for the current shell –

[root@rhel7 ~]# ps

PID TTY TIME CMD

12330 pts/0 00:00:00 bash

21621 pts/0 00:00:00 ps

Result contains four columns of information.Where:  
**PID –** the unique process ID  
**TTY –** terminal type that the user is logged into  
**TIME –** amount of CPU in minutes and seconds that the process has been running  
**CMD –** name of the command that launched the process.

**Note –** Sometimes when we execute **ps** command, it shows TIME as 00:00:00. It is nothing but the total accumulated CPU utilization time for any process and 00:00:00 indicates no CPU time has been given by the kernel till now. In above example we found that, for bash no CPU time has been given. This is because bash is just a parent process for different processes which needs bash for their execution and bash itself is not utilizing any CPU time till now.

1. **View Processes :**View all the running processes use either of the following option with ps –

[root@rhel7 ~]# ps -A

[root@rhel7 ~]# ps -e

1. **View Processes not associated with a terminal:** View all processes except both session leaders and processes not associated with a terminal.

[root@rhel7 ~]# ps -a

PID TTY TIME CMD

27011 pts/0 00:00:00 man

27016 pts/0 00:00:00 less

27499 pts/1 00:00:00 ps

**Note –** You may be thinking that what is session leader? A unique session is assing to evry process group. So, session leader is a process which kicks off other processes. The process ID of first process of any session is similar as the session ID.

1. **View all the processes except session leaders :**

[root@rhel7 ~]# ps -d

1. **View all processes except those that fulfill the specified conditions (negates the selection) :**  
   *Example –* If you want to see only session leader and processes not associated with a terminal. Then, run

[root@rhel7 ~]# ps -a -N

OR

[root@rhel7 ~]# ps -a --deselect

1. **View all processes associated with this terminal :**

[root@rhel7 ~]# ps -T

1. **View all the running processes :**

[root@rhel7 ~]# ps -r

1. **View all processes owned by you :**Processes i.e same EUID as ps which means runner of the ps command, root in this case –

[root@rhel7 ~]# ps -x

### **Process selection by list**

Here we will discuss how to get the specific processes list with the help of *ps* command. These options accept a single argument in the form of a blank-separated or comma-separated list. They can be used multiple times.  
*For example:* ps -p “1 2” -p 3, 4

1. Select the process by the command name. This selects the processes whose executable name is given in cmdlist. There may be a chance you won’t know the process ID and with this command it is easier to search.  
   **Syntax :** ps -C command\_name

Syntax :

ps -C command\_name

Example :

[root@rhel7 ~]# ps -C dhclient

PID TTY TIME CMD

19805 ? 00:00:00 dhclient

1. Select by group ID or name. The group ID identifies the group of the user who created the process.

Syntax :

ps -G group\_name

ps --Group group\_name

Example :

[root@rhel7 ~]# ps -G root

1. View by group id :

Syntax :

ps -g group\_id

ps -group group\_id

Example :

[root@rhel7 ~]# ps -g 1

PID TTY TIME CMD

1 ? 00:00:13 systemd

1. View process by process ID.

Syntax :

ps p process\_id

ps -p process\_id

ps --pid process\_id

Example :

[root@rhel7 ~]# ps p 27223

PID TTY STAT TIME COMMAND

27223 ? Ss 0:01 sshd: root@pts/2

[root@rhel7 ~]# ps -p 27223

PID TTY TIME CMD

27223 ? 00:00:01 sshd

[root@rhel7 ~]# ps --pid 27223

PID TTY TIME CMD

27223 ? 00:00:01 sshd

You can view multiple processes by specifying multiple process IDs separated by blank or comma –  
*Example :*

[root@rhel7 ~]# ps -p 1 904 27223

PID TTY STAT TIME COMMAND

1 ? Ss 0:13 /usr/lib/systemd/systemd --switched-root --system --d

904 tty1 Ssl+ 1:02 /usr/bin/X -core -noreset :0 -seat seat0 -auth /var/r

27223 ? Ss 0:01 sshd: root@pts/2

Here, we mentioned three process IDs – 1, 904 and 27223 which are separated by blank.

1. Select by parent process ID. By using this command we can view all the processes owned by parent process except the parent process.

[root@rhel7 ~]# ps -p 766

PID TTY TIME CMD

766 ? 00:00:06 NetworkManager

[root@rhel7 ~]# ps --ppid 766

PID TTY TIME CMD

19805 ? 00:00:00 dhclient

In above example process ID **766** is assigned to NetworkManager and this is the parent process for dhclient with process ID 19805.

1. View all the processes belongs to any session ID.

Syntax :

ps -s session\_id

ps --sid session\_id

Example :

[root@rhel7 ~]# ps -s 1248

PID TTY TIME CMD

1248 ? 00:00:00 dbus-daemon

1276 ? 00:00:00 dconf-service

1302 ? 00:00:00 gvfsd

1310 ? 00:00:00 gvfsd-fuse

1369 ? 00:00:00 gvfs-udisks2-vo

1400 ? 00:00:00 gvfsd-trash

1418 ? 00:00:00 gvfs-mtp-volume

1432 ? 00:00:00 gvfs-gphoto2-vo

1437 ? 00:00:00 gvfs-afc-volume

1447 ? 00:00:00 wnck-applet

1453 ? 00:00:00 notification-ar

1454 ? 00:00:02 clock-applet

1. Select by tty. This selects the processes associated with the mentioned tty :

Syntax :

ps t tty

ps -t tty

ps --tty tty

Example :

[root@rhel7 ~]# ps -t pts/0

PID TTY TIME CMD

31199 pts/0 00:00:00 bash

31275 pts/0 00:00:00 man

31280 pts/0 00:00:00 less

1. Select by effective user ID or name.  
   *Syntax :*  
   ps U user\_name/ID  
   ps -U user\_name/ID  
   ps -u user\_name/ID  
   ps –User user\_name/ID  
   ps –user user\_name/ID

### **Output Format control**

These options are used to choose the information displayed by ps. There are multiple options to control output format. These option can be combined with any other options like **e, u, p, G, g** etc, depends on our need.

* 1. Use **-f** to view full-format listing.

[tux@rhel7 ~]$ ps -af

tux 17327 17326 0 12:42 pts/0 00:00:00 -bash

tux 17918 17327 0 12:50 pts/0 00:00:00 ps -af

* 1. Use **-F** to view Extra full format.

[tux@rhel7 ~]$ ps -F

UID PID PPID C SZ RSS PSR STIME TTY TIME CMD

tux 17327 17326 0 28848 2040 0 12:42 pts/0 00:00:00 -bash

tux 17942 17327 0 37766 1784 0 12:50 pts/0 00:00:00 ps -F

* 1. To view process according to user-defined format.

Syntax :

[root@rhel7 ~]# ps --formate column\_name

[root@rhel7 ~]# ps -o column\_name

[root@rhel7 ~]# ps o column\_name

Example :

[root@rhel7 ~]# ps -aN --format cmd,pid,user,ppid

CMD PID USER PPID

/usr/lib/systemd/systemd -- 1 root 0

[kthreadd] 2 root 0

[ksoftirqd/0] 3 root 2

[kworker/0:0H] 5 root 2

[migration/0] 7 root 2

[rcu\_bh] 8 root 2

[rcu\_sched] 9 root 2

[watchdog/0] 10 root 2

In this example I wish to see command, process ID, username and parent process ID, so I pass the arguments cmd, pid, user and ppid respectively.

* 1. View in BSD job control format :

[root@rhel7 ~]# ps -j

PID PGID SID TTY TIME CMD

16373 16373 16373 pts/0 00:00:00 bash

19734 19734 16373 pts/0 00:00:00 ps

* 1. Display BSD long format :

[root@rhel7 ~]# ps l

F UID PID PPID PRI NI VSZ RSS WCHAN STAT TTY TIME COMMAND

4 0 904 826 20 0 306560 51456 ep\_pol Ssl+ tty1 1:32 /usr/bin/X -core -noreset :0 -seat seat0 -auth /var/run/lightdm/root/:0 -noli

4 0 11692 11680 20 0 115524 2132 do\_wai Ss pts/2 0:00 -bash

* 1. Add a column of security data.

[root@rhel7 ~]# ps -aM

LABEL PID TTY TIME CMD

unconfined\_u:unconfined\_r:unconfined\_t:s0-s0:c0.c1023 19534 pts/2 00:00:00 man

unconfined\_u:unconfined\_r:unconfined\_t:s0-s0:c0.c1023 19543 pts/2 00:00:00 less

unconfined\_u:unconfined\_r:unconfined\_t:s0-s0:c0.c1023 20469 pts/0 00:00:00 ps

View command with signal format.

[root@rhel7 ~]# ps s 766

* 1. Display user-oriented format

[root@rhel7 ~]# ps u 1

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

root 1 0.0 0.6 128168 6844 ? Ss Apr08 0:16 /usr/lib/systemd/systemd --switched-root --system --deserialize 21

* 1. Display virtual memory format

[root@rhel7 ~]# ps v 1

PID TTY STAT TIME MAJFL TRS DRS RSS %MEM COMMAND

1 ? Ss 0:16 62 1317 126850 6844 0.6 /usr/lib/systemd/systemd --switched-root --system --deserialize 21

* 1. If you want to see environment of any command. Then use option \*\*e\*\* –

[root@rhel7 ~]# ps ev 766

PID TTY STAT TIME MAJFL TRS DRS RSS %MEM COMMAND

766 ? Ssl 0:08 47 2441 545694 10448 1.0 /usr/sbin/NetworkManager --no-daemon LANG=en\_US.UTF-8 PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin

* 1. View processes using highest memory.

ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem

* 1. print a process tree

[root@rhel7 ~]# ps --forest -C sshd

PID TTY TIME CMD

797 ? 00:00:00 sshd

11680 ? 00:00:03 \\_ sshd

16361 ? 00:00:02 \\_ sshd

* 1. List all threads for a particular process. Use either the **-T or -L** option to display threads of a process.

[root@rhel7 ~]# ps -C sshd -L

PID LWP TTY TIME CMD

797 797 ? 00:00:00 sshd

11680 11680 ? 00:00:03 sshd

16361 16361 ? 00:00:02 sshd

**Note –** For the explanation of different column contents refer **man page.**

# 30 Useful ‘ps Command’ Examples for Linux Process Monitoring

**ps** (**processes status**) is a native Unix/Linux utility for viewing information concerning a selection of running processes on a system: it reads this information from the virtual files in [/proc filesystem](https://www.tecmint.com/exploring-proc-file-system-in-linux/). It is one of the important utilities for system administration specifically under process monitoring, to help you understand whats is going on a Linux system.

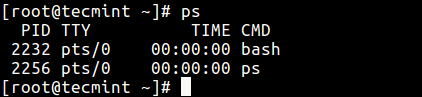
It has numerous options for manipulating its output, however you’ll find a small number of them practically useful for daily usage.

**Read Also**: [All You Need To Know About Processes in Linux [Comprehensive Guide]](https://www.tecmint.com/linux-process-management/)

In this article, we’ll look at 30 useful examples of ps commands for monitoring active running processes on a Linux system.

Note that **ps** produces output with a heading line, which represents the meaning of each column of information, you can find the meaning of all the labels in the **ps man page**.

### **1) List All Processes in Current Shell**

1. If you run **ps command** without any arguments, it displays processes for the current shell.

$ ps

*List Current Running Processes*

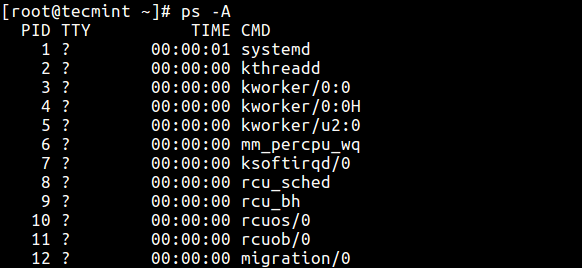
### **2) Print All Processes in Different Formats**

1. Display every active process on a Linux system in generic (Unix/Linux) format.

$ ps -A

OR

$ ps -e



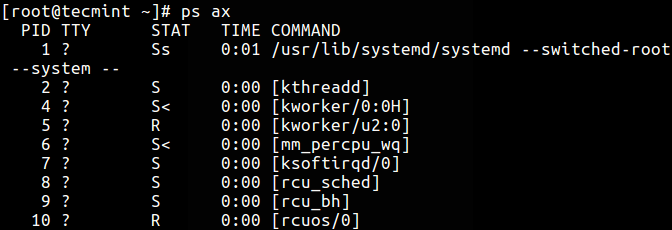
*List Processes in Standard Format*

1. Display all processes in BSD format.

$ ps au

OR

$ ps axu



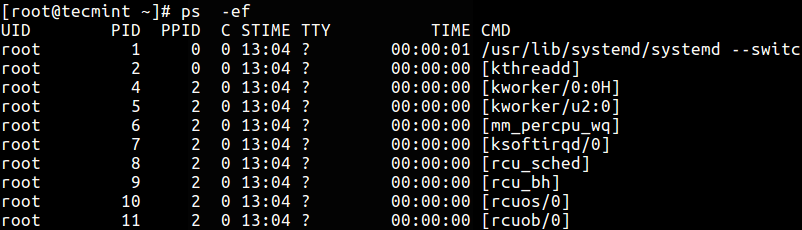
*List Processes in BSD Format*

1. To perform a full-format listing, add the -f or -F flag.

$ ps -ef

OR

$ ps -eF



*List Processes in Long List Format*

### **3) Display User Running Processes**

1. You can select all processes owned by you (runner of the **ps command**, root in this case), type:

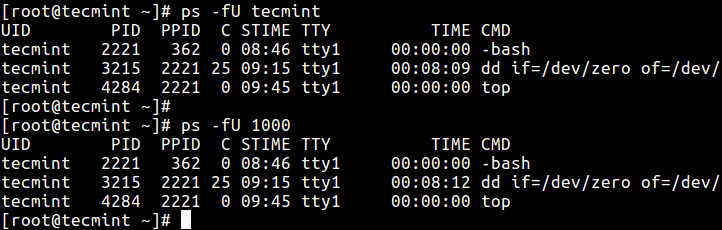
$ ps -x

1. To display a user’s processes by real user ID (**RUID**) or name, use the **-U** flag.

$ ps -fU tecmint

OR

$ ps -fu 1000



*List User Processes by ID*

1. To select a user’s processes by effective user **ID** (**EUID**) or name, use the **-u** option.

$ ps -fu tecmint

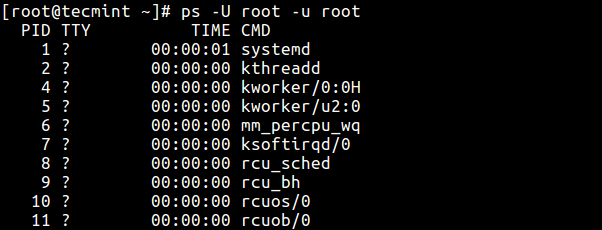
OR

$ ps -fu 1000

### **4) Print All Processes Running as Root (Real and Effecitve ID)**

**8.** The command below enables you to view every process running with **root** user privileges (real & effective ID) in user format.

$ ps -U root -u root



*Display Root User Running Processes*

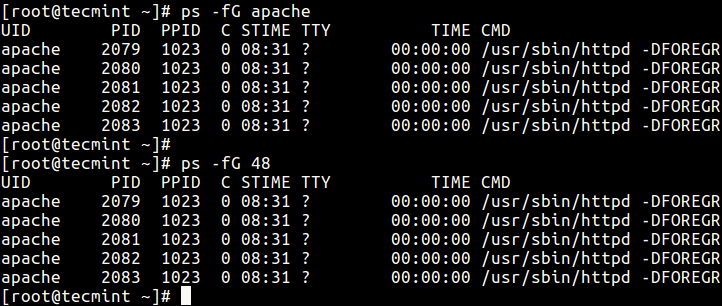
### **5) Display Group Processes**

**9.** If you want to list all processes owned by a certain group (real group ID (**RGID**) or name), type.

$ ps -fG apache

OR

$ ps -fG 48



*Display Group Processes*

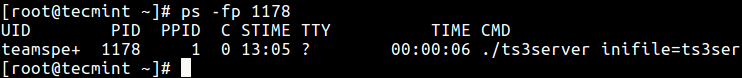
**10.** To list all processes owned by effective group name (or session), type.

$ ps -fg apache

### **6) Display Processes by PID and PPID**

**11.** You can list processes by **PID** as follows.

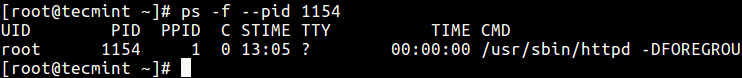
$ ps -fp 1178



*List Processes by PID*

**12.** To select process by **PPID**, type.

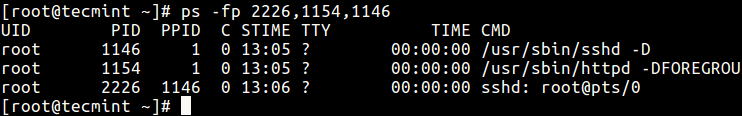
$ ps -f --ppid 1154



*List Process by PPID*

**13.** Make selection using **PID** list.

$ ps -fp 2226,1154,1146



*List Processes by PIDs*

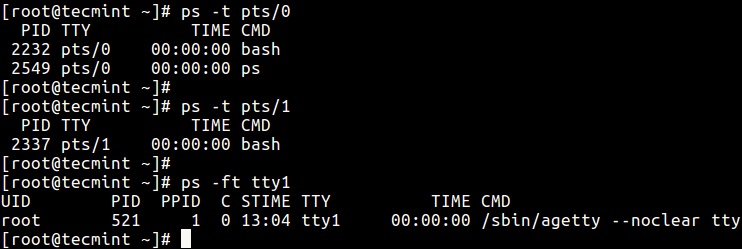
### **7) Display Processes by TTY**

**14.** To select processes by **tty**, use the **-t** flag as follows.

$ ps -t pst/0

$ ps -t pst/1

$ ps -ft tty1

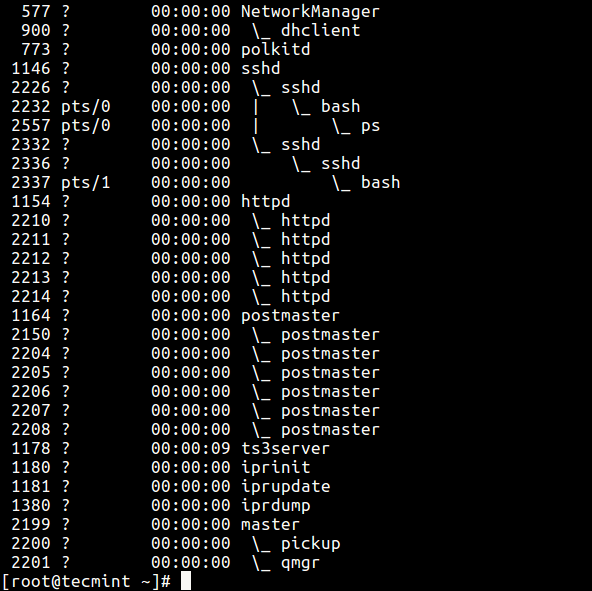


*List Processes by TTY*

### **8) Print Process Tree**

**15.** A process tree shows how processes on the system are linked to each other; processes whose parents have been killed are adopted by the init (or systemd).

$ ps -e --forest



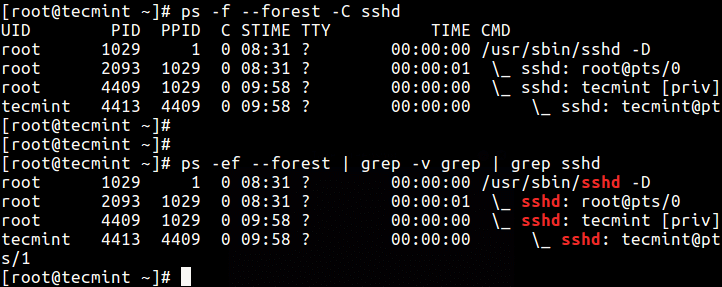
*List Process Tree*

**16.** You can also print a process tree for a given process like this.

$ ps -f --forest -C sshd

OR

$ ps -ef --forest | grep -v grep | grep sshd

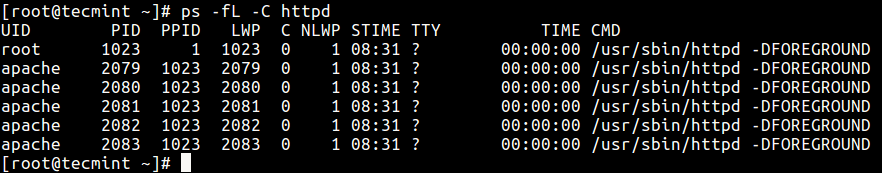


*List Tree View of Process*

### **9) Print Process Threads**

**17.** To print all threads of a process, use the **-H** flag, this will show the **LWP** (**light weight process**) as well as **NLWP** (**number of light weight process**) columns.

$ ps -fL -C httpd



*List Process Threads*

### **10) Specify Custom Output Format**

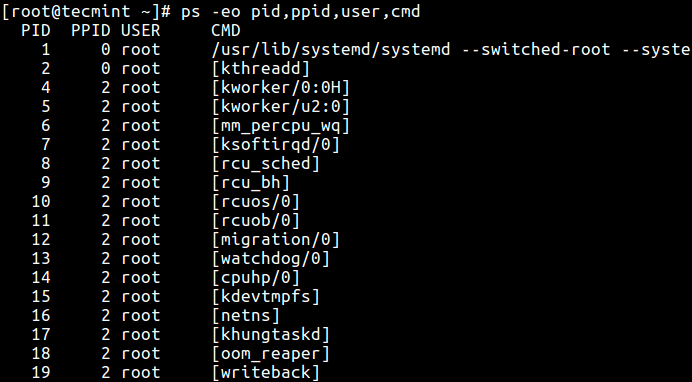
Using the **-o** or **–format** options, ps allows you to build user-defined output formats as shown below.

**18.** To list all format specifiers, include the **L** flag.

$ ps L

**19.** The command below allows you to view the **PID**, **PPID**, user name and command of a process.

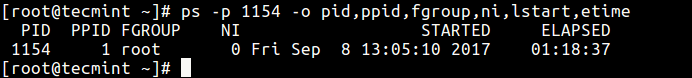
$ ps -eo pid,ppid,user,cmd



*List Processes with Names*

**20.** Below is another example of a custom output format showing file system group, nice value, start time and elapsed time of a process.

$ ps -p 1154 -o pid,ppid,fgroup,ni,lstart,etime



*List Process ID Information*

**21.** To find a [process name using its PID](https://www.tecmint.com/find-process-name-pid-number-linux/).

$ ps -p 1154 -o comm=

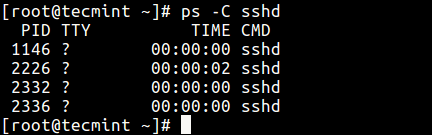
Find Process using PID

*Find Process using PID*

### **11) Display Parent and Child Processes**

**22.** To select a specific process by its name, use the -C flag, this will also display all its child processes.

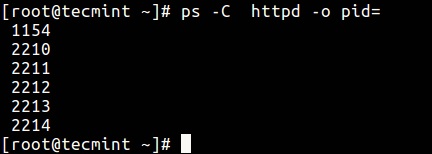
$ ps -C sshd



*Find Parent Child Process*

**23.** Find all **PIDs** of all instances of a process, useful when writing scripts that need to read **PIDs** from a std output or file.

$ ps -C httpd -o pid=

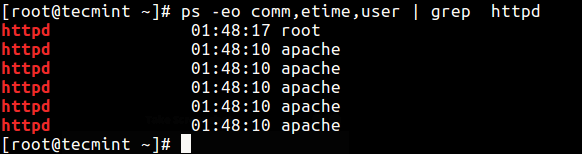


*Find All Process PIDs*

**24.** Check execution time of a process.

$ ps -eo comm,etime,user | grep httpd

The output below shows the HTTPD service has been running for 1 hours, 48 minutes and 17 seconds.



*Find Process Uptime*

### **12) Troubleshoot Linux System Performance**

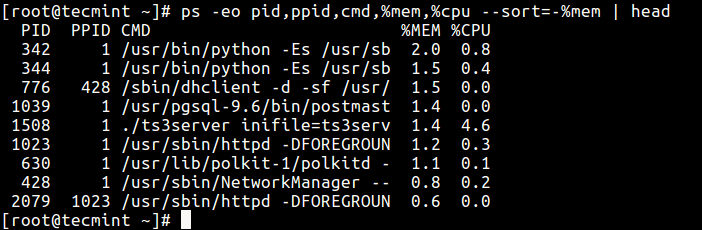
If your system isn’t working as it should be, for instance if it’s unusually slow, you can [perform some system troubleshooting](https://www.tecmint.com/linux-system-monitoring-troubleshooting-tools/) as follows.

**26.** Find [top running processes](https://www.tecmint.com/find-linux-processes-memory-ram-cpu-usage/) by highest memory and CPU usage in Linux.

$ ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem | head

OR

$ ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%cpu | head



*Find Top Running Processes*

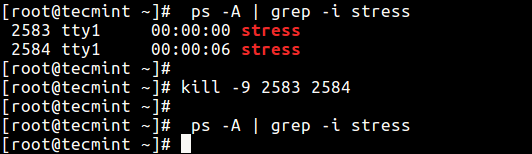
**27.** To kill an Linux [processes/unresponsive applications](https://www.tecmint.com/kill-processes-unresponsive-programs-in-ubuntu/) or any process that is consuming high CPU time.

First, find the **PID** of the unresponsive process or application.

$ ps -A | grep -i stress

Then use the [kill command](https://www.tecmint.com/how-to-kill-a-process-in-linux/) to terminate it immediately.

$ kill -9 2583 2584



*Find and Kill a Process*

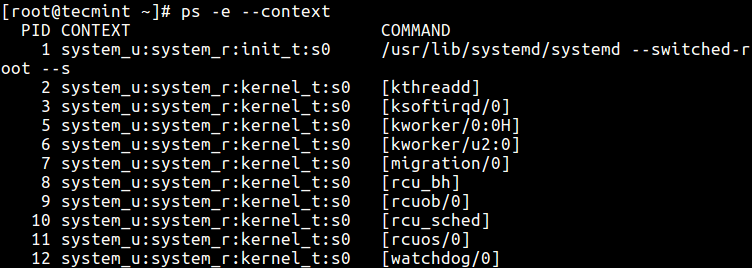
### **13) Print Security Information**

**28.** Show security context (specifically for **SELinux**) like this.

$ ps -eM

OR

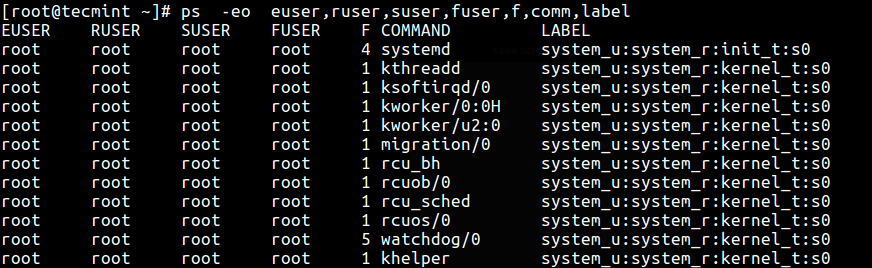
$ ps --context



*Find SELinux Context*

**29.** You can also display security information in user-defined format with this command.

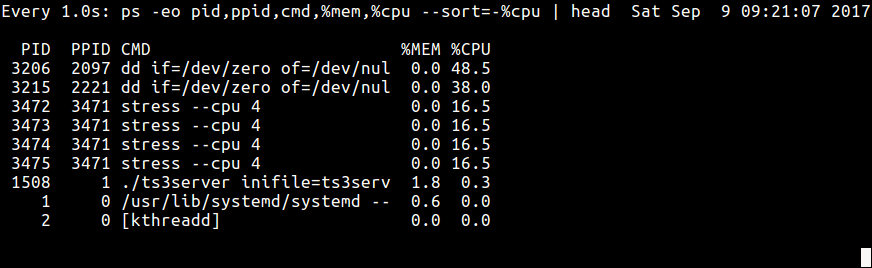
$ ps -eo euser,ruser,suser,fuser,f,comm,label



*List SELinux Context by Users*

### **14) Perform Real-time Process Monitoring Using Watch Utility**

**30.** Finally, since **ps** displays static information, you can employ the [watch utility](https://www.tecmint.com/fswatch-monitors-files-and-directory-changes-modifications-in-linux/) to perform real-time process monitoring with repetitive output, displayed after every second as in the command below (specify a custom **ps command** to achieve your objective).

$ watch -n 1 'ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem | head'

*Real Time Process Monitoring*

**Important**: ps only shows static information, to view frequently updated output you can use tools such as [htop](https://www.tecmint.com/install-htop-linux-process-monitoring-for-rhel-centos-fedora/" \t "_blank); [top](https://www.tecmint.com/12-top-command-examples-in-linux/) and [glances](https://www.tecmint.com/glances-an-advanced-real-time-system-monitoring-tool-for-linux/): the last two are in fact Linux system performance monitoring tool.

You might also like to read these following related articles.

1. [How to Find a Process Name Using PID Number in Linux](https://www.tecmint.com/find-process-name-pid-number-linux/)
2. [Find Top Running Processes by Highest Memory and CPU Usage in Linux](https://www.tecmint.com/find-linux-processes-memory-ram-cpu-usage/)
3. [A Guide to Kill, Pkill and Killall Commands to Terminate a Process in Linux](https://www.tecmint.com/how-to-kill-a-process-in-linux/)
4. [How to Find and Kill Running Processes in Linux](https://www.tecmint.com/find-and-kill-running-processes-pid-in-linux/)
5. [How to Start Linux Command in Background and Detach Process in Terminal](https://www.tecmint.com/run-linux-command-process-in-background-detach-process/)

That’s all for now. If you have any useful **ps command** example(s) to share (not forgetting to explain what it does), use the comment form below.

Sharing is Caring...

# All You Need To Know About Processes in Linux [Comprehensive Guide]

In this article, we will walk through a basic understanding of processes and briefly look at[*how to manage processes in Linux*](https://www.tecmint.com/dstat-monitor-linux-server-performance-process-memory-network/) using certain commands.

A **process** refers to a program in execution; it’s a running instance of a program. It is made up of the program instruction, data read from files, other programs or input from a system user.

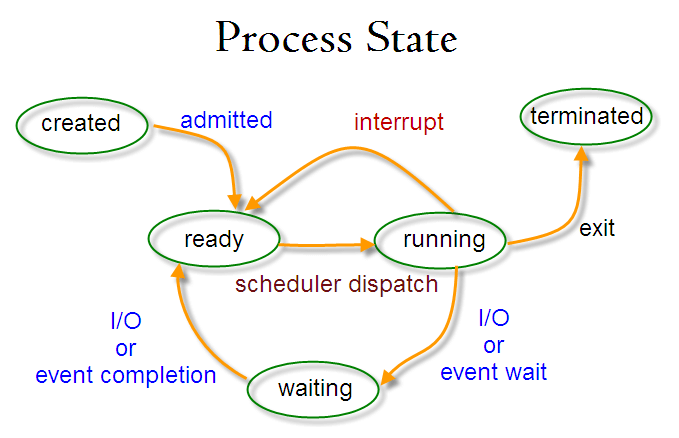
#### **Types of Processes**

There are fundamentally two types of processes in Linux:

* ***Foreground processes*** (also referred to as interactive processes) – these are initialized and controlled through a terminal session. In other words, there has to be a user connected to the system to start such processes; they haven’t started automatically as part of the system functions/services.
* ***Background processes*** (also referred to as non-interactive/automatic processes) – are processes not connected to a terminal; they don’t expect any user input.

#### **What is Daemons**

These are special types of background processes that start at system startup and keep running forever as a service; they don’t die. They are started as system tasks (run as services), spontaneously. However, they can be controlled by a user via the init process.

[](https://www.tecmint.com/wp-content/uploads/2017/03/ProcessState.png)

*Linux Process State*

### **Creation of a Processes in Linux**

A new process is normally created when an existing process makes an exact copy of itself in memory. The child process will have the same environment as its parent, but only the process ID number is different.

There are two conventional ways used for creating a new process in Linux:

* **Using The system() Function** – this method is relatively simple, however, it’s inefficient and has significantly certain security risks.
* **Using fork() and exec() Function** – this technique is a little advanced but offers greater flexibility, speed, together with security.

### **How Does Linux Identify Processes?**

Because Linux is a multi-user system, meaning different users can be running various programs on the system, each running instance of a program must be identified uniquely by the kernel.

And a program is identified by its **process ID (PID)** as well as its parent processes ID (**PPID**), therefore processes can further be categorized into:

* **Parent processes** – these are processes that create other processes during run-time.
* **Child processes** – these processes are created by other processes during run-time.

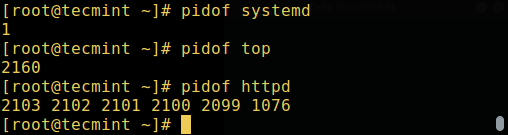
#### **The Init Process**

***Init*** process is the mother (parent) of all processes on the system, it’s the first program that is executed when the [Linux system boots up](https://www.tecmint.com/linux-boot-process/); it manages all other processes on the system. It is started by the kernel itself, so in principle it does not have a parent process.

The init process always has process ID of **1**. It functions as an adoptive parent for all orphaned processes.

You can use the ***pidof* command** to find the ID of a process:

# pidof systemd

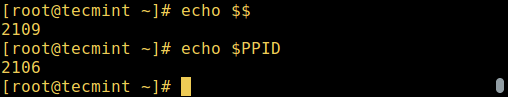
[](https://www.tecmint.com/wp-content/uploads/2017/03/Find-Linux-Process-ID.png)# pidof top

# pidof httpd

*Find Linux Process ID*

To find the process ID and parent process ID of the current shell, run:

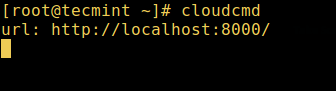
$ echo $$

[](https://www.tecmint.com/wp-content/uploads/2017/03/Find-Linux-Parent-Process-ID.png)$ echo $PPID

*Find Linux Parent Process ID*

#### **Starting a Process in Linux**

Once you run a command or program (for example cloudcmd – CloudCommander), it will start a process in the system. You can start a foreground (interactive) process as follows, it will be connected to the terminal and a user can send input it:

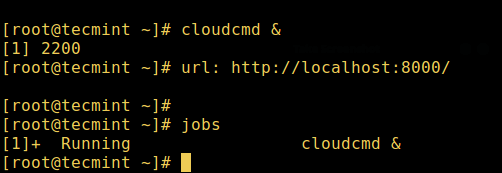
[](https://www.tecmint.com/wp-content/uploads/2017/03/Start-Linux-Interactive-Process.png)# cloudcmd

*Start Linux Interactive Process*

#### **Linux Background Jobs**

To start a process in the background (non-interactive), use the **&** symbol, here, the process doesn’t read input from a user until it’s moved to the foreground.

# cloudcmd &

[](https://www.tecmint.com/wp-content/uploads/2017/03/Start-Linux-Process-in-Background.png)# jobs

*Start Linux Process in Background*

You can also send a process to the background by suspending it using **[Ctrl + Z]**, this will send the **SIGSTOP** signal to the process, thus stopping its operations; it becomes idle:

# tar -cf backup.tar /backups/\* **#press Ctrl+Z**

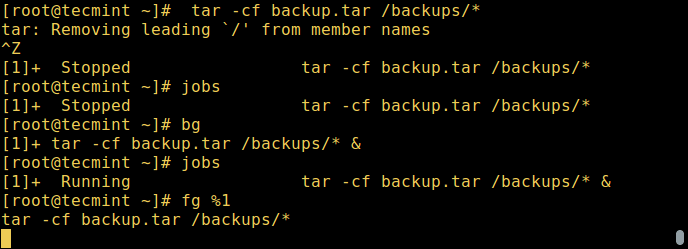
# jobs

To continue running the above-suspended command in the background, use the **bg** command:

# bg

To send a background process to the foreground, use the **fg** command together with the job ID like so:

# jobs

[](https://www.tecmint.com/wp-content/uploads/2017/03/Linux-Background-Process-Jobs.png)# fg %1

*Linux Background Process Jobs*

You may also like: [How to Start Linux Command in Background and Detach Process in Terminal](https://www.tecmint.com/run-linux-command-process-in-background-detach-process/)

#### **States of a Process in Linux**

During execution, a process changes from one state to another depending on its environment/circumstances. In Linux, a process has the following possible states:

* ***Running***– here it’s either running (it is the current process in the system) or it’s ready to run (it’s waiting to be assigned to one of the CPUs).
* ***Waiting*** – in this state, a process is waiting for an event to occur or for a system resource. Additionally, the kernel also differentiates between **two types of waiting processes**; interruptible waiting processes – can be interrupted by signals and uninterruptible waiting processes – are waiting directly on hardware conditions and cannot be interrupted by any event/signal.
* ***Stopped***– in this state, a process has been stopped, usually by receiving a signal. For instance, a process that is being debugged.
* ***Zombie*** – here, a process is dead, it has been halted but it’s still has an entry in the process table.

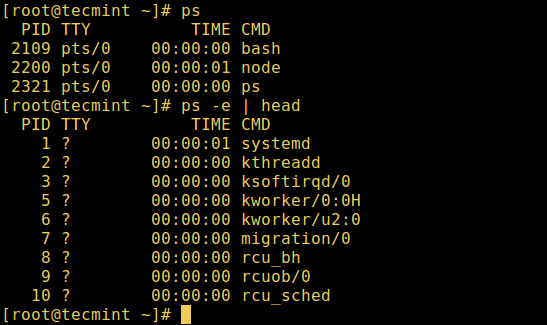
#### **How to View Active Processes in Linux**

There are several Linux tools for viewing/listing running processes on the system, the two traditional and well known are [ps](https://www.tecmint.com/linux-boot-process-and-manage-services/) and [top](https://www.tecmint.com/12-top-command-examples-in-linux/) commands:

#### **a) ps Command**

It displays information about a selection of the active processes on the system as shown below:

# ps

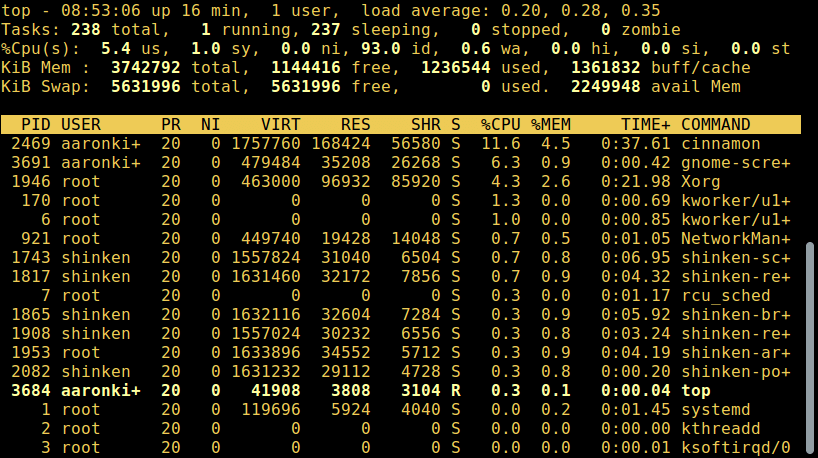
[](https://www.tecmint.com/wp-content/uploads/2017/03/ps-command.png)# ps -e | head

*List Linux Active Processes*

#### **b) top – System Monitoring Tool**

[*top is a powerful tool*](https://www.tecmint.com/12-top-command-examples-in-linux/) that offers you a [*dynamic real-time view of a running system*](https://www.tecmint.com/bcc-best-linux-performance-monitoring-tools/) as shown in the screenshot below:

# top

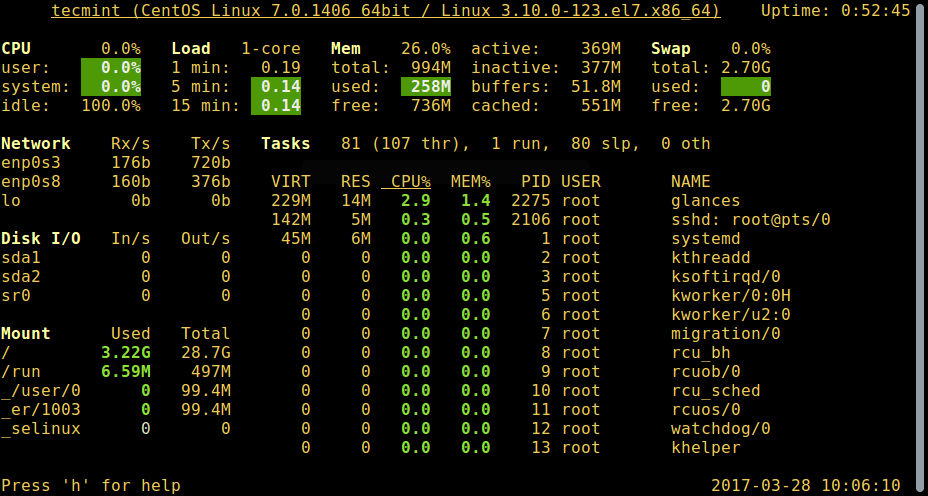
[](https://www.tecmint.com/wp-content/uploads/2017/03/top-command.png)

*List Linux Running Processes*

Read this for more top usage examples: [12 TOP Command Examples in Linux](https://www.tecmint.com/12-top-command-examples-in-linux/)

#### **c) glances – System Monitoring Tool**

**glances** is a relatively new system monitoring tool with advanced features:

[](https://www.tecmint.com/wp-content/uploads/2017/03/glances.png)# glances

*Glances – Linux Process Monitoring*

For a comprehensive usage guide, read through: [Glances – An Advanced Real Time System Monitoring Tool for Linux](https://www.tecmint.com/glances-an-advanced-real-time-system-monitoring-tool-for-linux/)

There are several other useful Linux system monitoring tools you can use to list active processes, open the link below to read more about them:

[20 Command Line Tools to Monitor Linux Performance](https://www.tecmint.com/command-line-tools-to-monitor-linux-performance/)

[13 More Useful Linux Monitoring Tools](https://www.tecmint.com/linux-performance-monitoring-tools/)

### **How to Control Processes in Linux**

Linux also has some commands for controlling processes such as kill, pkill, pgrep and killall, below are a few basic examples of how to use them:

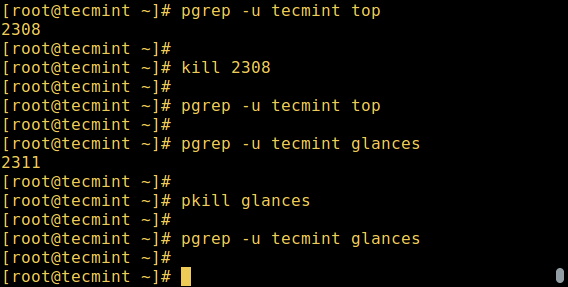
$ pgrep -u tecmint top

$ kill 2308

$ pgrep -u tecmint top

$ pgrep -u tecmint glances

$ pkill glances

[](https://www.tecmint.com/wp-content/uploads/2017/03/Control-Linux-Processes.png)$ pgrep -u tecmint glances

*Control Linux Processes*

To learn how to use these commands in-depth, to kill/terminate active processes in Linux, open the links below:

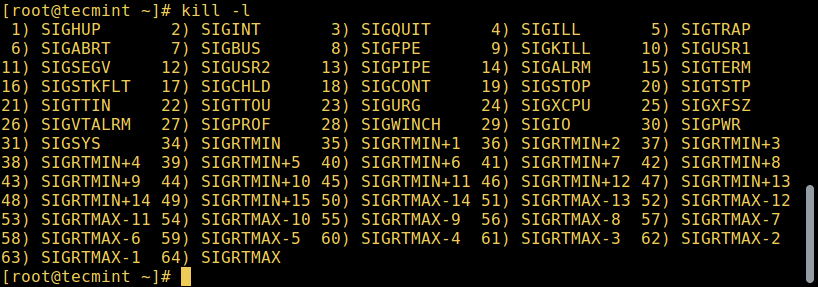
[A Guide to Kill, Pkill and Killall Commands to Terminate Linux Processess](https://www.tecmint.com/how-to-kill-a-process-in-linux/)

[How to Find and Kill Running Processes in Linux](https://www.tecmint.com/find-and-kill-running-processes-pid-in-linux/)

Note that you can use them to kill [unresponsive applications in Linux](https://www.tecmint.com/kill-processes-unresponsive-programs-in-ubuntu/) when your system freezes.

#### **Sending Signals To Processes**

The fundamental way of controlling processes in Linux is by sending signals to them. There are multiple signals that you can send to a process, to view all the signals run:

[](https://www.tecmint.com/wp-content/uploads/2017/03/list-all-signals.png)$ kill -l

*List All Linux Signals*

To send a signal to a process, use the kill, pkill or pgrep commands we mentioned earlier on. But programs can only respond to signals if they are programmed to recognize those signals.

And most signals are for internal use by the system, or for programmers when they write code. The following are signals which are useful to a system user:

* **SIGHUP 1** – sent to a process when its controlling terminal is closed.
* **SIGINT 2** – sent to a process by its controlling terminal when a user interrupts the process by pressing **[Ctrl+C]**.
* **SIGQUIT 3** – sent to a process if the user sends a quit signal **[Ctrl+D]**.
* **SIGKILL 9** – this signal immediately terminates (kills) a process and the process will not perform any clean-up operations.
* **SIGTERM 15** – this a program termination signal (kill will send this by default).
* **SIGTSTP 20** – sent to a process by its controlling terminal to request it to stop (terminal stop); initiated by the user pressing **[Ctrl+Z]**.

The following are kill commands examples to kill the Firefox application using its PID once it freezes:

$ pidof firefox

$ kill 9 2687

OR

$ kill -KILL 2687

OR

$ kill -SIGKILL 2687

To kill an application using its name, use pkill or killall like so:

$ pkill firefox

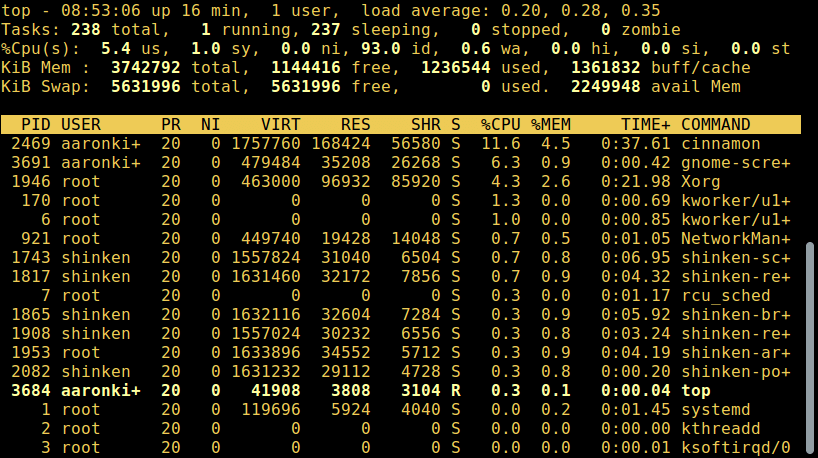
$ killall firefox

#### **Changing Linux Process Priority**

On the Linux system, all active processes have a priority and certain nice value. Processes with higher priority will normally get more CPU time than lower priority processes.

However, a system user with root privileges can influence this with the **nice** and **renice** commands.

From the output of the top command, the NI shows the process nice value:

[](https://www.tecmint.com/wp-content/uploads/2017/03/top-command.png)$ top

*List Linux Running Processes*

Use the **nice** command to set a nice value for a process. Keep in mind that normal users can attribute a nice value from zero to 20 to processes they own.  
Only the root user can use negative nice values.

To **renice** the priority of a process, use the **renice** command as follows:

$ renice +8 2687

$ renice +8 2103

Check out our some useful articles on how to manage and control Linux processes.

[Linux Process Management: Boot, Shutdown, and Everything in Between](https://www.tecmint.com/rhcsa-exam-boot-process-and-process-management/)

[Find Top 15 Processes by Memory Usage with ‘top’ in Batch Mode](https://www.tecmint.com/find-processes-by-memory-usage-top-batch-mode/)

[Find Top Running Processes by Highest Memory and CPU Usage in Linux](https://www.tecmint.com/find-linux-processes-memory-ram-cpu-usage/)

[How to Find a Process Name Using PID Number in Linux](https://www.tecmint.com/find-process-name-pid-number-linux/)

That’s all for now! Do you have any questions or additional ideas, share them with us via the feedback form below.

Sharing is Caring...

ps - report a snapshot of the current processes.

SYNOPSIS

ps [options]

DESCRIPTION

ps displays information about a selection of the active processes. If you want a repetitive update of the selection and the displayed information, use top(1) instead.

This version of ps accepts several kinds of options:

1. UNIX options, which may be grouped and must be preceded by a dash.
2. BSD options, which may be grouped and must not be used with a dash.
3. GNU long options, which are preceded by two dashes.

Options of different types may be freely mixed, but conflicts can appear. There are some synonymous options, which are functionally identical, due to the many standards and ps implementations that this ps is compatible with.

Note that "ps -aux" is distinct from "ps aux". The POSIX and UNIX standards require that "ps -aux" print all processes owned by a user named "x", as well as printing all processes that would be selected by the -a option. If the user named "x" does not exist, this ps may interpret the command as "ps aux" instead and print a warning. This behavior is intended to aid in transitioning old scripts and habits. It is fragile, subject to change, and thus should not be relied upon.

By default, ps selects all processes with the same effective user ID (euid=EUID) as the current user and associated with the same terminal as the invoker. It displays the process ID (pid=PID), the terminal associated with the process (tname=TTY), the cumulated CPU time in [DD-]hh:mm:ss format (time=TIME), and the executable name (ucmd=CMD). Output is unsorted by default.

The use of BSD-style options will add process state (stat=STAT) to the default display and show the command args (args=COMMAND) instead of the executable name. You can override this with the PS\_FORMAT environment variable. The use of BSD-style options will also change the process selection to include processes on other terminals (TTYs) that are owned by you; alternately, this may be described as setting the selection to be the set of all processes filtered to exclude processes owned by other users or not on a terminal. These effects are not considered when options are described as being "identical" below, so -M will be considered identical to Z and so on.

Except as described below, process selection options are additive. The default selection is discarded, and then the selected processes are added to the set of processes to be displayed. A process will thus be shown if it meets any of the given selection criteria.