FEBRUARY 3, 2023 / #LINUX

# How to Manage Linux Processes

Arunachalam B

We all follow certain processes to achieve our goals. Similarly, every system has its own processes to accomplish tasks.

Every program or command that executes in a Linux system is called a process.

In this tutorial, let's explore processes and how we can manage them in Linux.

## What is a Linux Process?

A process is theoretically called a program in execution. It's basically a task that a system is currently working on.

Every action you take on the system will result in a new process. For example, opening a browser initiates a process.

In simple words, a process is an instance of a program. The user action is transformed into a command and a new process will be

of the hierarchy implies, a process initiated from a command/program is called the parent process and the produced process of a parent process is called the child process.

# Types of Linux Processes

Processes are classified into 2 types in Linux Distributions:

1. Foreground Processes
2. Background Processes

## Foreground processes

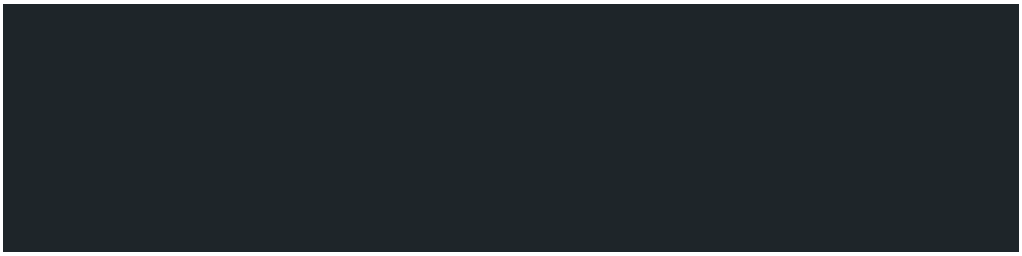
A process that requires the user to start it using a Terminal command or Program is called a foreground process. This means that foreground processes require an input trigger from a user. So every foreground process is manually triggered.

Whenever a process is running in the foreground, the other processes should wait until the current process completes.

The best example to demonstrate this is via the `sleep` command. The `sleep` command does not allow the user to interact with the terminal until a given number of seconds has passed.

```
sleep 10
```

Terminal command to sleep for 10 sec in foreground



`sleep` terminal command running on foreground and blocks user input

We should wait for 10 seconds to access the terminal to run another command.

## Background Processes

A process that runs independently on user input is called a background process. Unlike the foreground processes, we can run multiple processes at the same time in a background process.

To run a process in the background, place an ampersand (&) at the end of the command that you use to start the process.

Here's a quick example to demonstrate that:

Let's execute the `sleep` command in a background process. It'll run in the background and gives the terminal back to us to run other commands.


Sample Terminal command for a background process

```
sleep 10 &
```

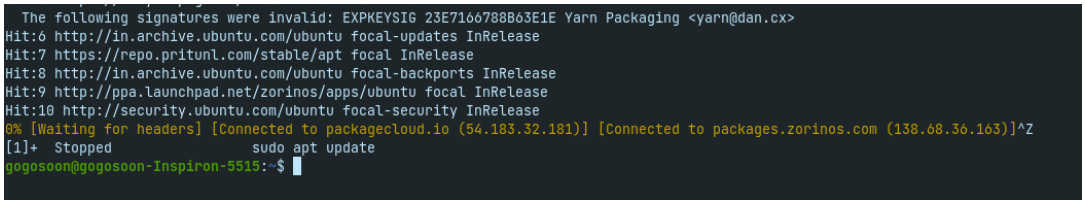
Terminal command to sleep for 10 sec in background.

Now we can see that the above command runs in the background. It created a process with the `PID` ( 19003 ). So we can run another command simultaneously ( `pwd` command).

# How to Change a Foreground Process to a Background Process

If we start a process in the foreground and would like to place it in the background, we can do it using the `bg` command. Let's see how to change the foreground process to the background.

If a process is running, press the key `CTRL+Z` . This command will suspend the current process.




Foreground process output

Then run the `bg` command. It takes a process id as an argument and places the process into the background. If the argument is empty it will place the currently suspended process in the background.

```
bg <process_id>
```

Command to move the process to background

```
bg
```

Command to move the last process to background


Foreground process to background process output

# How to List Linux Processes

Before we go over how to do this, you should know why you might need to know a list of processes. Here are a few reasons:

1. To know which process consumes more time.
2. To know which process takes more memory and CPU usage.
3. To know the triggered command for a running process.

To see the processes that are running currently, we can use `ps` (Process Status) command:

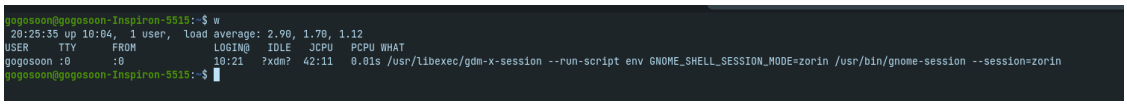
```
ps
```

Terminal command to list the running processes


`ps` command showing the list of running processes

`w`

Terminal command to list the processes of all logged in users

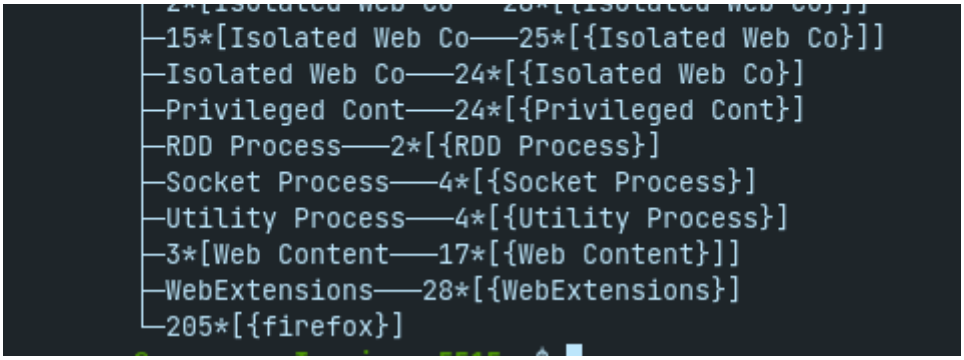



`w` command displaying the list of processes of all users

# How to List the Processes in Tree View

When a program/command runs, it initiates a main process called the parent process. The parent process may depend on some other command/program which will create a child process.

Here's an example screenshot.




Child processes of the parent (firefox) process

In the above screenshot, Firefox is the parent process and the other processes are its child processes.

Let's explore how to list the process in a tree-like structure.

`pstree` is a Linux command to list the currently running process of all users in a tree-like structure. It is used as a more visual alternative to the `ps` command.

```
pstree
```

Terminal command to list the processes in a tree like structure


`pstree` command listing processes in tree view

As we can see, the running processes are in tree form. This can be useful to visualize the processes.

Adding the `-p` flag with the command will display each branch with its process id.

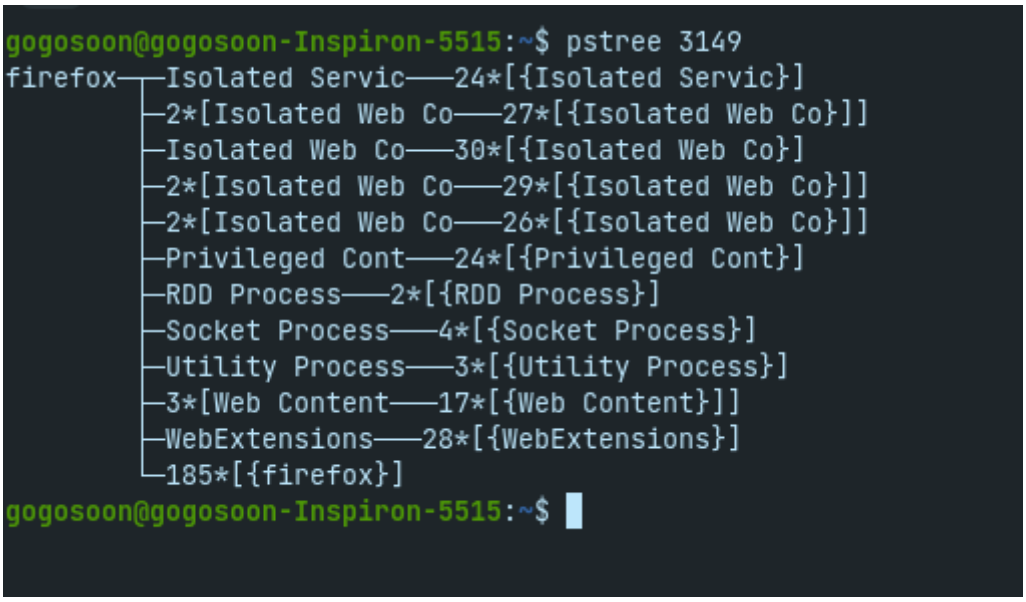
```
pstree -p
```

Terminal command to display the child processes of a parent process in tree like structure


Terminal command displaying the list of processes in tree view with PID

```
pstree 3149
```

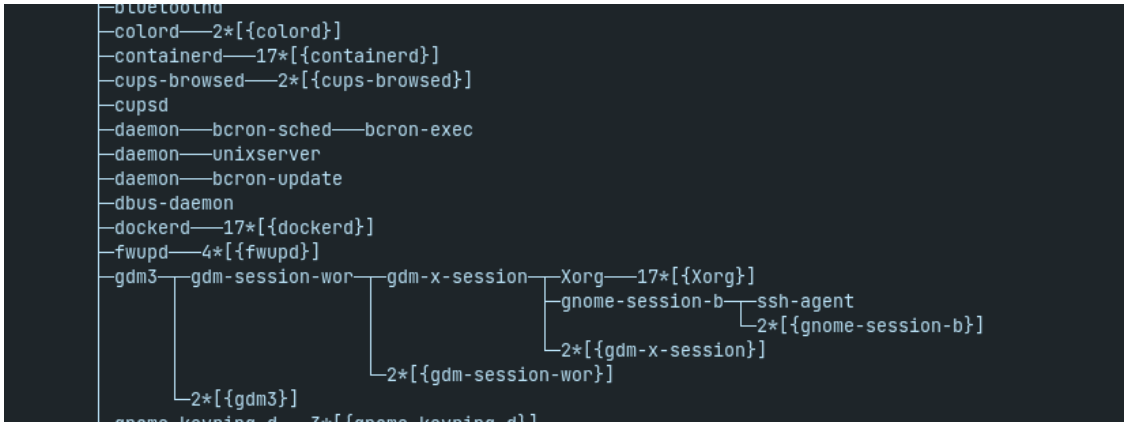
Terminal command to list the processes of 3149 process id




Listing processes in tree view for a particular process

Earlier, I mentioned that `pstree` command lists the processes from all the users. Passing the username along with the `pstree` command lists only the processes run by the user.

```
pstree root
```




Listing processes in tree view for a particular user

The above screenshot shows the processes running by the `root` user.

# How to See the Processes of a Particular Program

Many developers may have faced the following scenario:

While working on web development projects, we use browsers like Chrome, Firefox, and others to verify the output with different browsers.

Some developers will keep on opening the tabs and never close the opened ones. Due to heavy load (if 150+ tabs are opened) browsers will not respond sometimes 😣 leading to the system hanging up. The worst part could be that we won't be able to close the browser 😂.

Unlike Windows, we don't have Task Manager in Linux to kill the browser.

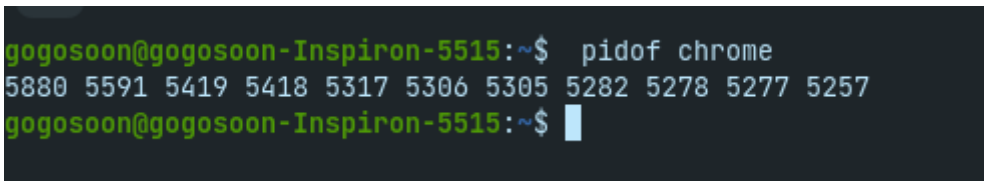
We know that every program (including the browser) runs as a process. So then you just have to find the process id and kill it.

Let's see how to find the process id of a command/program you need.

In my system, Chrome is running, Now we can get the PIDs of Chrome by running the following command:

```
pidof chrome
```

Terminal Command to find ID of a process




Terminal command to find process id of chrome

# How to Kill a Process

There is a command called `kill` in Linux that is used to kill any process by passing the PID ( Process id ) or Process Name.

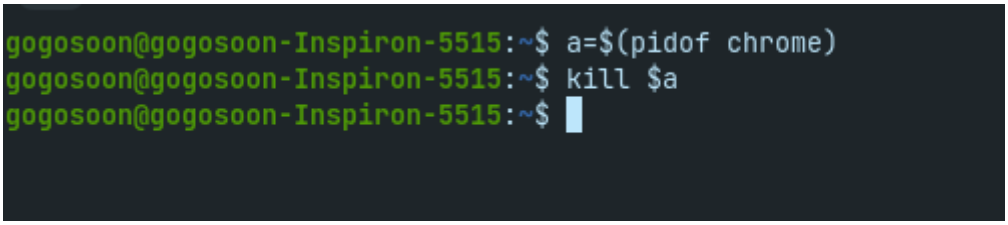
Here's the syntax of the `kill` command:

Syntax for kill command

Let's store the PID of Chrome and kill it using the kill command:

```
a=$(pidof chrome)
kill $a
```

Command to kill a process




Terminal command to kill a process

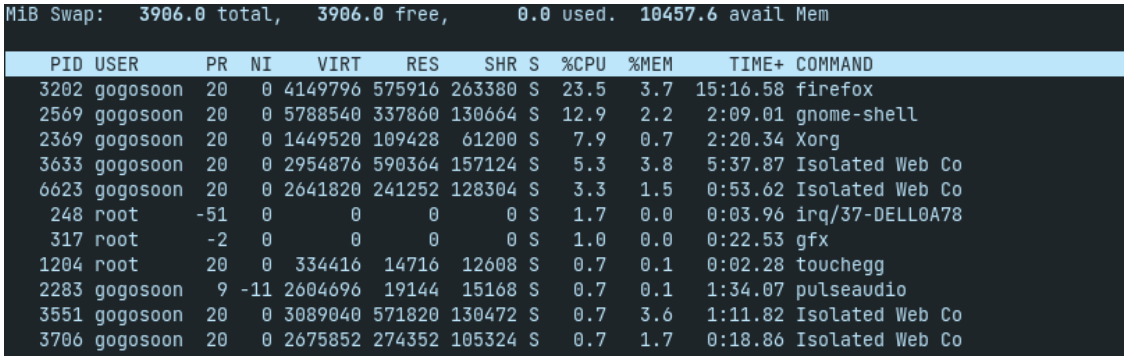
The above command will kill the Chrome web browser.

# How to List All Processes

We can see all the Linux processes using the `top` command. It shows real-time updates of each process for all users.

```
top
```

Terminal command to list all the processes in real-time




Terminal command displaying all the process in real-time

Let's understand the heading to understand the underlying data.

- PID represents a Unique process ID.
- USER represents the Username of the owner of the task.
- PR represents the Priority of the process. Lower the number, higher the priority.
- NI represents a Nice Value of task. A Negative nice value implies higher priority, and a positive Nice value means lower priority.
- VIRT represents the total virtual memory used by the task.
- RES represents RAM Usage of a process in kilobytes.
- SHR represents Shared Memory Size ( Kb ) used by a process.
- S represents the Status of the process:
  - **D:** Uninterruptible sleep
  - **R:** Running
  - **S:** Sleeping
  - **T:** Traced (stopped)
  - **Z:** Zombie
- CPU represents the CPU usage.

- COMMAND represents the command that used to start the process.

To display specific user processes we should use the flag `-u` :

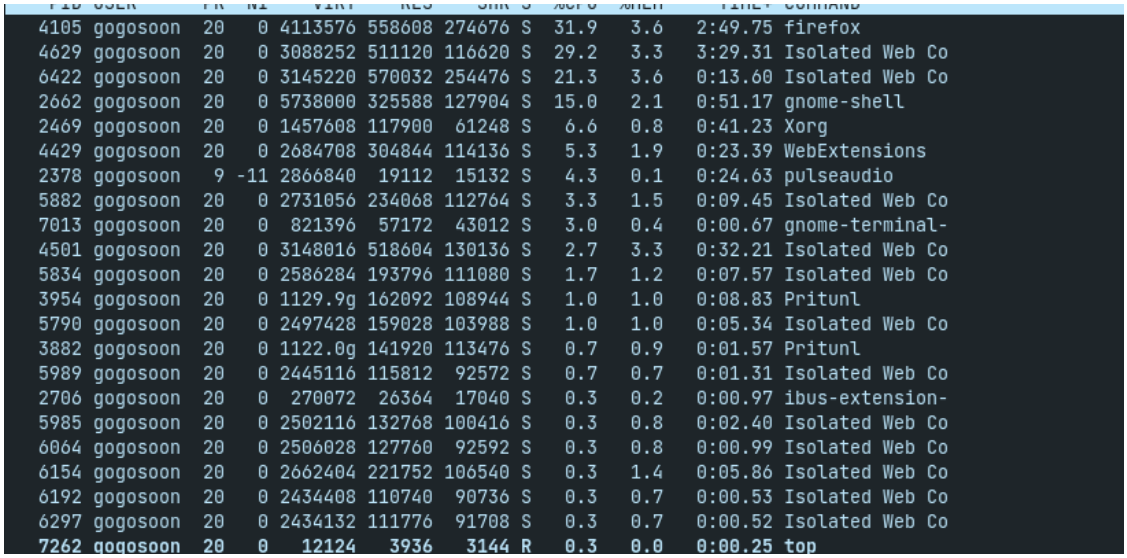
```
top -u <username>
```

Terminal command syntax to list specific user's processes

To look at the processes run by the user `gogosoon` , run the following command:

```
top -u gogosoon
```

Terminal command to list processes started by user `gogosoon`



```
  4105 gogosoon  20   0 4113576 558608 274676 S  31.9   3.6   2:49.75 firefox
  4629 gogosoon  20   0 3088252 511120 116620 S  29.2   3.3   3:29.31 Isolated Web Co
  6422 gogosoon  20   0 3145220 570032 254476 S  21.3   3.6   0:13.60 Isolated Web Co
  2662 gogosoon  20   0 5738000 325588 127904 S  15.0   2.1   0:51.17 gnome-shell
  2469 gogosoon  20   0 1457608 117900  61248 S   6.6   0.8   0:41.23 Xorg
  4429 gogosoon  20   0 2684708 304844 114136 S   5.3   1.9   0:23.39 WebExtensions
  2378 gogosoon   9 -11 2866840  19112  15132 S   4.3   0.1   0:24.63 pulseaudio
  5882 gogosoon  20   0 2731056 234068 112764 S   3.3   1.5   0:09.45 Isolated Web Co
  7013 gogosoon  20   0  821396  57172  43012 S   3.0   0.4   0:00.67 gnome-terminal-
  4501 gogosoon  20   0 3148016 518604 130136 S   2.7   3.3   0:32.21 Isolated Web Co
  5834 gogosoon  20   0 2586284 193796 111080 S   1.7   1.2   0:07.57 Isolated Web Co
  3954 gogosoon  20   0 1129.9g 162092 108944 S   1.0   1.0   0:08.83 Pritunl
  5790 gogosoon  20   0 2497428 159028 103988 S   1.0   1.0   0:05.34 Isolated Web Co
  3882 gogosoon  20   0 1122.0g 141920 113476 S   0.7   0.9   0:01.57 Pritunl
  5989 gogosoon  20   0 2445116 115812  92572 S   0.7   0.7   0:01.31 Isolated Web Co
  2706 gogosoon  20   0  270072  26364  17040 S   0.3   0.2   0:00.97 ibus-extension-
  5985 gogosoon  20   0 2502116 132768 100416 S   0.3   0.8   0:02.40 Isolated Web Co
  6064 gogosoon  20   0 2506028 127760  92592 S   0.3   0.8   0:00.99 Isolated Web Co
  6154 gogosoon  20   0 2662404 221752 106540 S   0.3   1.4   0:05.86 Isolated Web Co
  6192 gogosoon  20   0 2434408 110740  90736 S   0.3   0.7   0:00.53 Isolated Web Co
  6297 gogosoon  20   0 2434132 111776  91708 S   0.3   0.7   0:00.52 Isolated Web Co
  7262 gogosoon  20   0   12124   3936   3144 R   0.3   0.0   0:00.25 top
```

Terminal output of all process started by user `gogosoon`

You might be confused about seeing the command line output 😆. It'll be a bit hard to debug the processes in real time.

Here comes the handy GUI tool to handle the processes in Linux. But we have to install this manually. This will work more like task manager in Windows.

```
sudo apt install gnome-system-monitor
```

Terminal command to install system monitoring app

After installing, just type the name of the software in terminal:

```
gnome-system-monitor
```

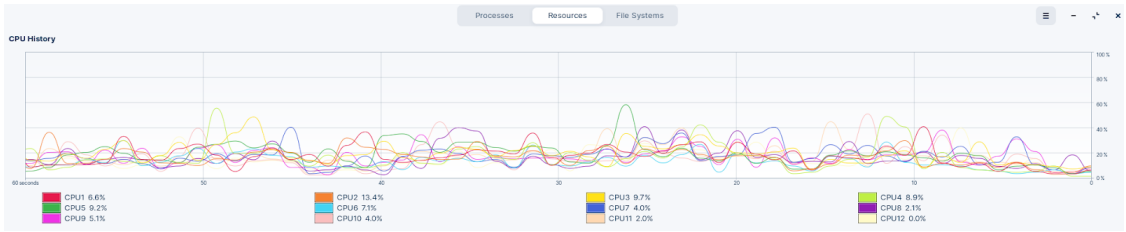
Command to open the List of process in GUI


Gnome-System-Monitor

When we right-click on any process it will show the actions like kill, stop, end, and so on.

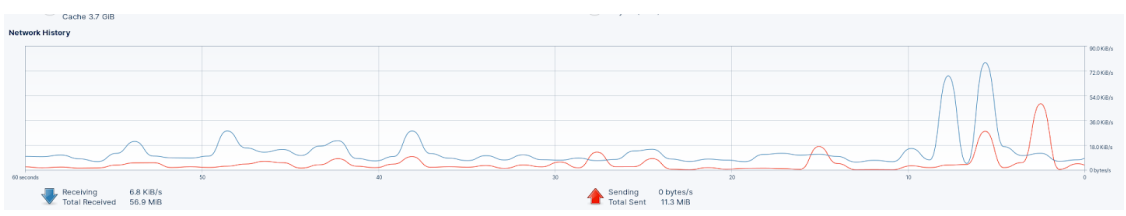
The Resources tab shows the following utilities:

1. CPU History
2. Memory and Swap History
3. Network History




CPU History graph


Memory and Swap History Graph



These graphs will be useful to determine the load in your system.

# Conclusion

In this article, you have learned the basics of processes in Linux. I hope you now understand how they work a bit better. I recommend you all try these commands in your system.

To learn more about Linux, subscribe to my email newsletter at my site and follow me on social media.

---

**Arunachalam B**

Project Manager by Profession. Software Developer by Passion. Curious to explore technologies. Live in Chennai

---

freeCodeCamp is a donor-supported tax-exempt 501(c)(3) charity organization (United States Federal Tax Identification Number: 82-0779546)

Our mission: to help people learn to code for free. We accomplish this by creating thousands of videos, articles, and interactive coding lessons - all freely available to the public. We also have thousands of freeCodeCamp study groups around the world.

Donations to freeCodeCamp go toward our education initiatives, and help pay for servers, services, and staff.

**You can make a tax-deductible donation here.**

**Trending Guides**

Python list .append()

Python .zip() Function

What is a QA Engineer?

Python Print Same Line

Rename Branches in Git

What Does Coding Mean?

What is Data Analysis?

How to Comment Out CSS

Double vs Float in C++

C++ Operators

Excel Formulas

Break in Python

Text Align in CSS

How to Minify CSS

Python Split String

Python List insert()

Merge Sort Algorithm

What is an SVG File?