How-To Geek

How to Kill Processes From the Linux Terminal

DAVE MCKAY @thegurkha
UPDATED MAY 28, 2019, 5:10PM EDT

Fatmawati Achmad Zaenuri/Shutterstock.com

Killing a process is sometimes the only way to get rid of it. Despite the harsh name, “killing” a process just means “forcing it to quit.” Here’s how to do it from the Linux or macOS command line.

What is a Process?

Running programs like your web browser, background processes associated with your desktop environment, and Linux system services are all processes.

You can lump processes into two groups:

- **Foreground processes** are ones that have been started or launched by a user. They may be in a terminal window, or they may be a graphical application.
- **Background processes** are all of the processes that are started automatically and don't have any interaction with users. They don't expect input from users nor do they present results or output to them. Background processes are things like services and daemons.

If the foreground processes are the front of theater staff and the actors, the background processes are the backstage "behind the scenes" team.

When processes misbehave or malfunction, they can hog too much CPU time, consume your RAM, or enter a tight computational loop and become unresponsive. Graphical applications can refuse to respond to mouse clicks. Terminal applications might never return you to the command prompt.

The Humane Answer

"Killing" a process just means "forcing the process to quit." This may be necessary if the process is refusing to respond.

Linux provides the `kill`, `pkill`, and `killall` commands to allow you to do just that. These commands can be used with any type of process, graphical or command line, foreground or background.

The kill Command

To use `kill`, you must know the process ID (PID) of the process you wish to terminate. The `ps` command can be used to find the PID of a process.

To have `ps` search through all of the processes use the `-e` (all processes) option. Piping the output through `less` is advisable, there's going to be quite a bit of it. Type `ps`, a space, `-e`, a space, `|` (a pipe character), another space and then type `less`. Press Enter to execute the command.

```
ps -e | less
```

This will give you a process listing that looks similar to the below screenshot. You can search forward in `less` using the `/` key and you can search backward using the `?` key.

To home in on the process you’re interested in, pipe the output from `ps` through `grep` and specify the name—or part of the name—of the process.

```
ps -e | grep shutter
```

Once you have located the PID of the process you wish to terminate, pass it to the `kill` command as a parameter. To terminate the `shutter` process identified by the previous command, use this command:

```
kill 2099
```

The `kill` command is a silent assassin—it does not give you any feedback if it was successful.

The pkill Command

The `pkill` command allows you to kill a process—or processes—by name. You do not need to identify the process by PID. To use `pkill` you provide a search term that `pkill` uses to check against the list of running processes. Matching processes are terminated. So you need to be positive you've got that search term spelled correctly.

As a safety net, you can use the `pgrep` command *before* you use the `pkill` command. The `pgrep` command also accepts a search term. It will list the PID of each process that matches the search term. This is safe because `pgrep` will not issue any kill signal to the processes, and if you mistype the search term you will not kill another process by mistake. You can make sure you have the search term correctly thought out before you pass it to `pkill`. Both `pkill` and `pgrep` treat the search term in the same way. Their treatment is so similar that they share the same man page.

Let's suppose there is a process with "subq" in its name. We'll use the `ps -u dave | grep` command to get a peek behind the curtain. You can see that "subq" will match that process and that process alone. That was just so you can see the full name of the process.

```
ps -u dave | grep subq
```

Let’s assume our user *hasn’t* done that; all they know is the process name contains the substring “subq.” They use `pgrep` to check that there is only one match to the search term. They then use that search term with `pkill`.

```
pgrep subq
```

```
pkill subq
```

You can use `pkill` to kill several processes at once. Here the user runs `pgrep` to check how many processes Chrome has launched. They use `pkill` to kill them all. They then check with `pgrep` that they have all been removed.

```
pgrep chrome
```

```
pkill chrome
```

```
pgrep chrome
```

If several processes with the same name are running, but you do not want to kill them all, you can use `pgrep` with the `-f` (command line) option to identify which process is which. A simple example would be two `ping` processes. You want to kill one of them but not the other. You can use their command lines to distinguish between them. Note the use of quotation marks to wrap the command line parameter.

```
pgrep -f "ping 192.168.4.22"
```

```
pkill -f "ping 192.168.4.22"
```

The killall Command

Warning: In the Solaris and OpenIndiana operating systems the `killall` command will *kill all the processes that belong to you*. If are root or if you have issued `sudo killall` you will reboot your computer! During the research for this article, this behavior was confirmed with the latest version of OpenIndiana Hipster 2018.10.

The `killall` command operates in a similar way to the `pkill` command but with a specific difference. Instead of passing a search term to the command you must provide the exact process name.

You cannot provide a partial match to a process name; you must provide the entire process name, as shown:

```
killall shutt
```

```
killall shutter
```

The `-y` (younger than) option allows you to kill processes that have been running for *less* than a specified period. The period is given in numbers followed by one of these units:

- s (seconds)
- m (minutes)
- h (hours)
- d (days)
- w (weeks)
- M (months, note, capital “M”)
- y (years)

To kill a process called `ana` that has just been launched and leave any older instances of `ana` running, you could use the following parameters with `killall`, if you’d reacted within two minutes:

```
killall -y 2m ana
```

The `-o` (older than) option allows you to kill processes that have been running for longer than a specified period. This command will kill all `ssh` connections that have been running for longer than a day:

```
killall -o 1d sshd
```

Don’t Be Too Trigger Happy

These commands will allow you to identify and terminate errant processes with accuracy and safety correctly.

Always be cautious. First, make sure the process you’re about to kill is really the one you want. Second, double check— be careful and ensure the targeted process is the one you want to end. Proceed with terminating the process once you’re satisfied.

READ NEXT

› How to Create Custom Widgets on iPhone

› How to Remember Where You Parked Using the Google Assistant

› How to Hide the Hidden Photos Album on iPhone and iPad

› How to Quickly Locate a Program’s EXE file on Windows 10

› How to Alternate Row or Column Shading in Google Sheets

DAVE MCKAY

Dave McKay first used computers when punched paper tape was in vogue, and he has been programming ever since. After over 30 years in the IT industry, he is now a full-time technology journalist. During his career, he has worked as a

freelance programmer, manager of an international software development team, an IT services project manager, and, most recently, as a Data Protection Officer. Dave is a Linux evangelist and open source advocate. **READ FULL BIO »**

The above article may contain affiliate links, which help support How-To Geek.