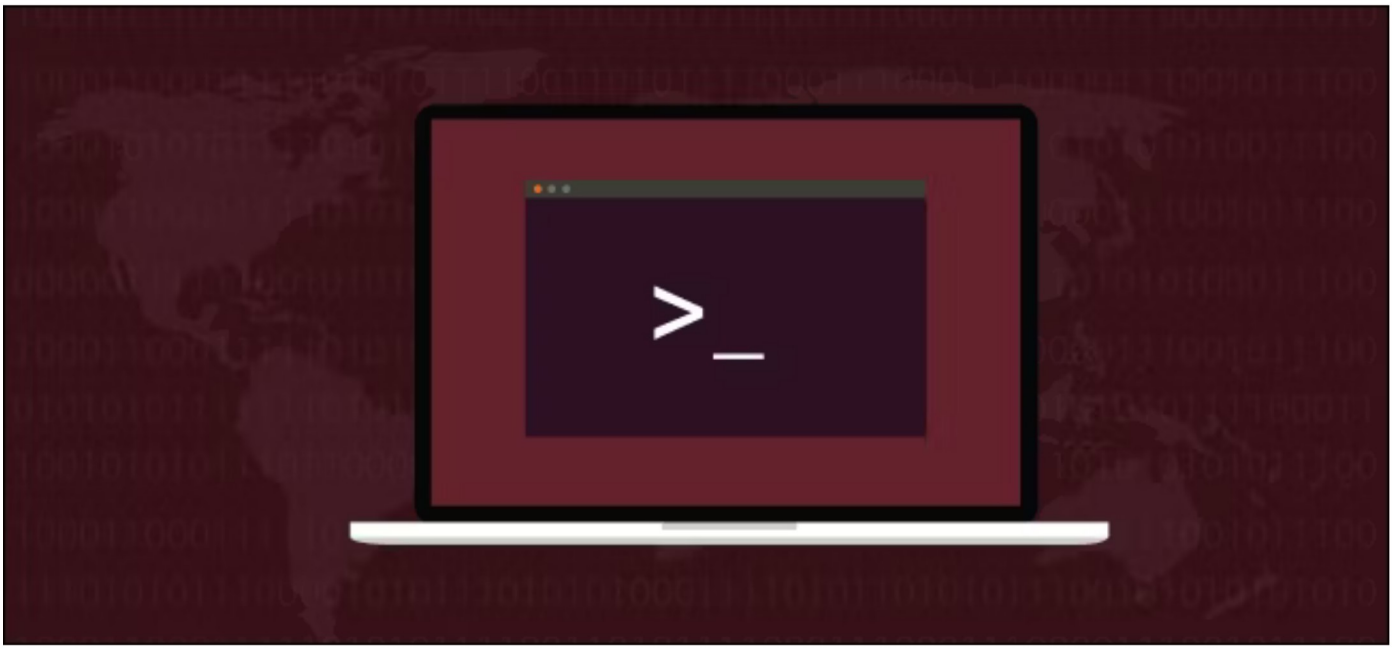
How to Use the ps Command to Monitor Linux Processes

Use Linux's ps command to find processes by name, user, or terminal, and find out what resources they're using, and more.

BY DAVE MCKAY PUBLISHED NOV 19, 2019



Quick Links

Get a snapshot of the processes running in your Linux computer with the `ps` command. Locate processes by name, user, or even terminal with as much or as little detail as you need. We show you how.

Process Management on Linux

The beating heart of all Linux and Unix-like operating systems is the kernel. Amongst its many responsibilities is the allocation of system resources such as RAM and CPU time. These have to be juggled in real-time so that all running processes get their fair share, according to the priority of each task.

Sometimes tasks can lock-up, or enter a tight loop, or become unresponsive for other reasons. Or they may continue running, but gobble up too much CPU time or RAM, or behave in some equally

anti-social way. Sometimes tasks need to be killed as a mercy to everyone involved. The first step. Of course, is to identify the process in question.

But maybe you don't have any task or performance issues at all. Perhaps you're just curious about which processes are running inside your computer, and you'd like to peek beneath the hood. The `ps` command satisfies both of these needs. It gives you a snapshot of what is happening inside your computer "right now."

`ps` is flexible enough to give you precisely the information you need in exactly the format you'd like it. In fact, `ps` has a great many options. The options described here will cater for most commonplace needs. If you need to go deeper into `ps` than we've taken it in this article, you'll find that our introduction makes the man page easier to digest.

Listing Processes

The easiest way to use `ps` is to fire it up with no parameters:

```
ps
```

`ps` displays a list of the processes started by the user who ran the command.

```
dave@howtogeek:~$ ps
  PID TTY          TIME CMD
21090 pts/0    00:00:00 bash
30626 pts/0    00:00:00 ps
dave@howtogeek:~$
```


The four columns are:

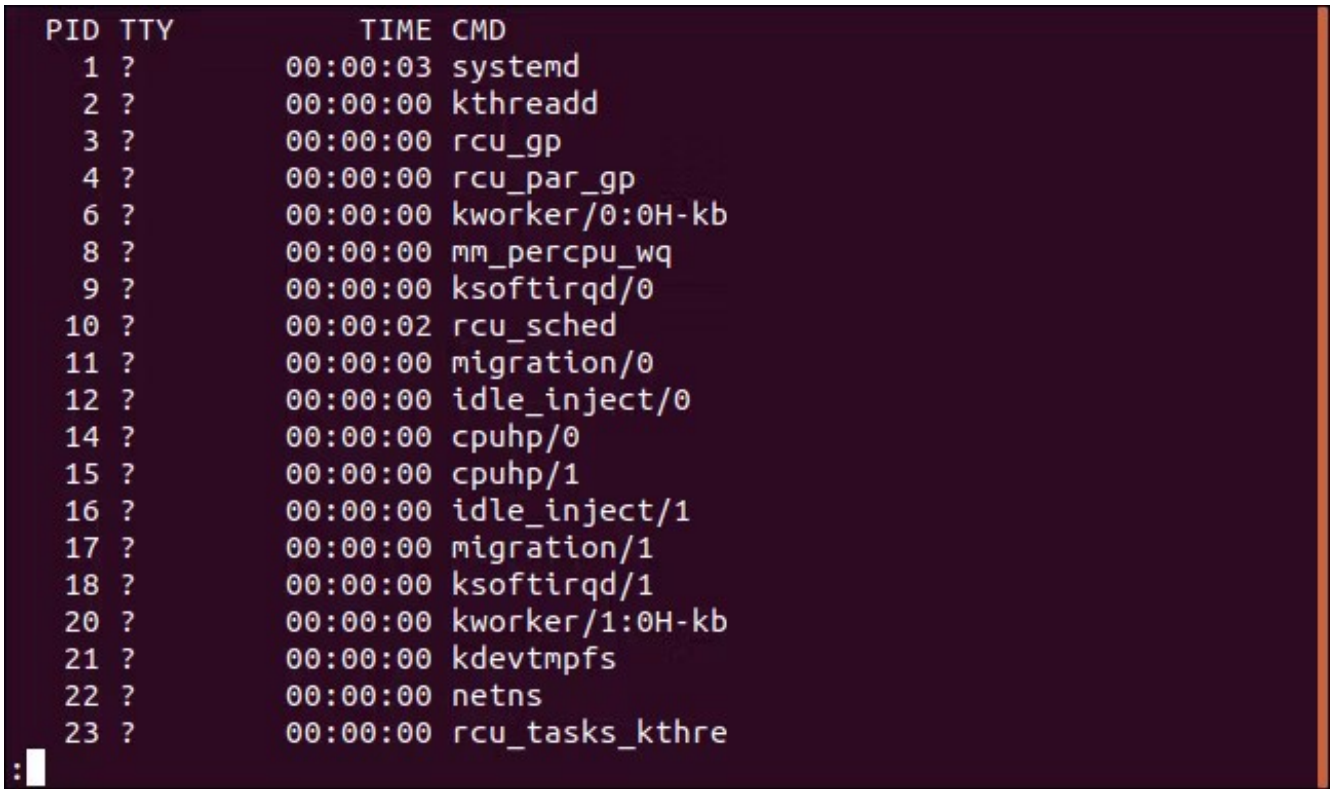
- **PID**: The process ID number of the process.
- **TTY**: The name of the console that the user is logged in at.
- **TIME**: The amount of CPU processing time that the process has used.
- **CMD**: The name of the command that launched the process

Listing Process for All Users

by adding the `-e` (select all processes) we can make `ps` list the processes that have been started by all users, not just the user who is running the `ps` command. Because this is going to be a long list, we're piping it into `less`.

```
ps -e | less
```

The process list is piped into `less`.



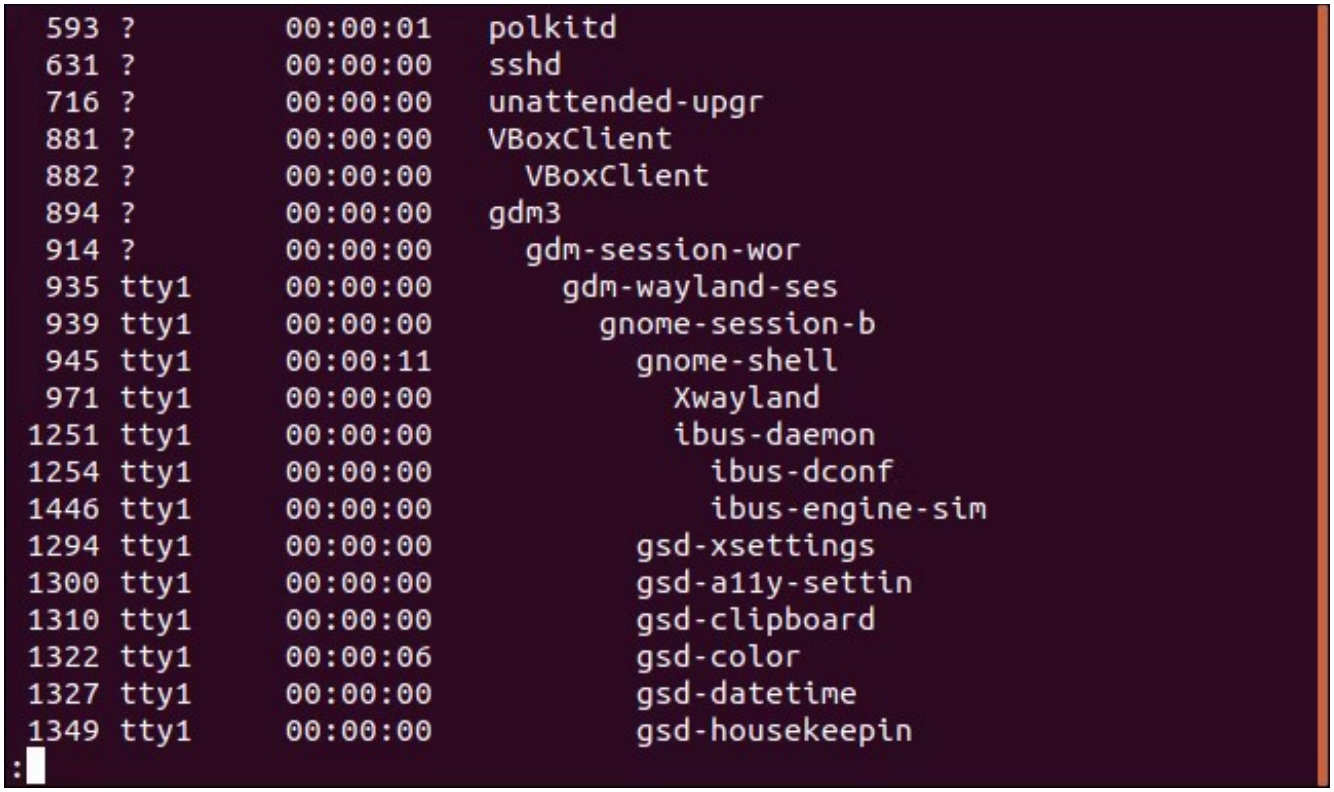
We've got many more entries in the list, but we see the same four columns as before. The entries with a question mark ? in the TTY column were not started from a terminal window.

Showing Process Hierarchy

Sometimes it can help to figure out an issue or identify a particular process if you can see which processes launched other processes. We use the -H (hierarchy) option to do so.

```
ps -eH | less
```

The indentation indicates which processes are parents of which other processes.



To add a little more clarity, we can ask `ps` to add some ASCII lines and to draw the hierarchy as a tree. The option to do this is the `--forest` option.

```
ps -eH --forest | less
```

```
dave@howtogeek:~$ ps -eH --forest | less
```

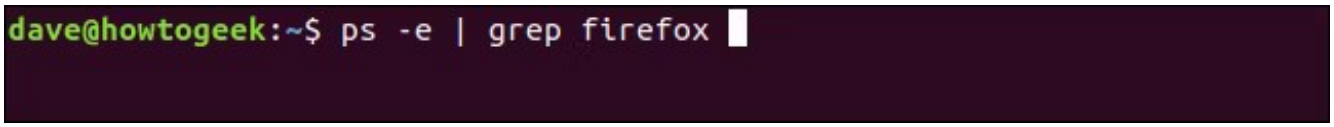
This makes it easier to track which processes are the parents of other processes.

```
  501 ?        00:00:00 accounts-daemon
  513 ?        00:00:00 networkd-dispat
  526 ?        00:00:00 udisksd
  532 ?        00:00:00 acpid
  593 ?        00:00:01 polkitd
  631 ?        00:00:00 sshd
  716 ?        00:00:00 unattended-upgr
  881 ?        00:00:00 VBoxClient
  882 ?        00:00:00  \_ VBoxClient
  894 ?        00:00:00 gdm3
  914 ?        00:00:00  \_ gdm-session-wor
  935 tty1     00:00:00  |   \_ gdm-wayland-ses
  939 tty1     00:00:00  |       \_ gnome-session-b
  945 tty1     00:00:11  |           \_ gnome-shell
  971 tty1     00:00:00  |           |   \_ Xwayland
 1251 tty1     00:00:00  |           |   \_ ibus-daemon
 1254 tty1     00:00:00  |           |       \_ ibus-dconf
 1446 tty1     00:00:00  |           |       \_ ibus-engine-sim
 1294 tty1     00:00:00  |           \_ gsd-xsettings
 1300 tty1     00:00:00  |           \_ gsd-a11y-settin
:
```

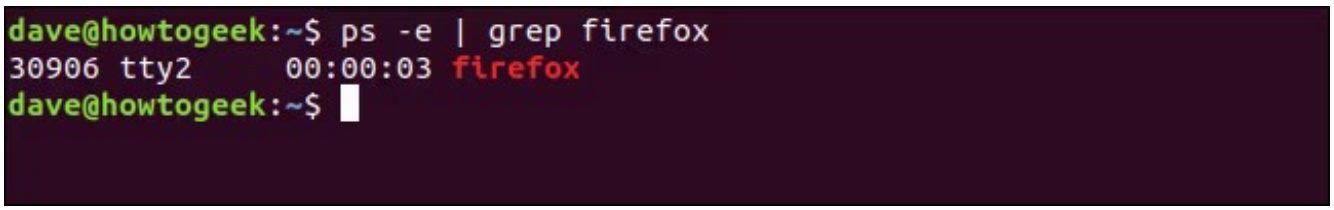
Listing Processes by Name

You can pipe the output from `ps` through `grep` to list entries that have names that match the search term. Here we're looking for entries that match the "firefox" search term:

```
ps -e | grep firefox
```



In this case, the output is a single entry for the process we are interested in. Of course, if we'd launched several instances of Firefox, there'd be more than one item returned in the list.



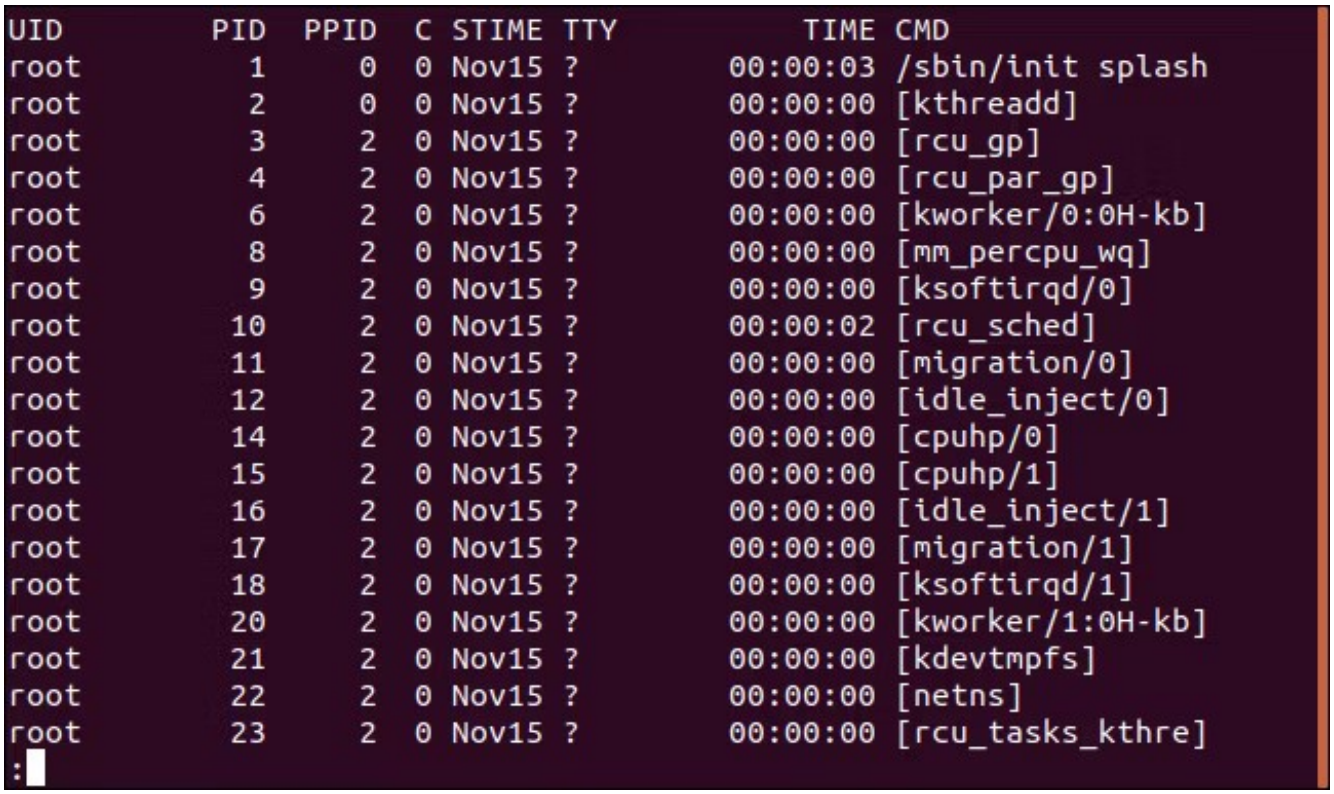
Showing More Columns in the Output

To add more columns to the output, use the `-f` (full-format) option.

```
ps -ef | less
```

ps -ef | less in a terminal window

An extra set of columns are included in the output from `ps`.



The columns are:

- **UID**: The user ID of the owner of this process.
- **PID**: The process ID of the process.
- **PPID**: Parent process ID of the process.
- **C**: The number of children the process has.
- **STIME**: Start time. The time when the process commenced.
- **TTY**: The name of the console that the user is logged in at.
- **TIME**: The amount of CPU processing time that the process has used.
- **CMD**: The name of the command that launched the process.

By using the `-F` (extra full-format) option we can get even more columns:

```
ps -eF | less
```

The columns we get this time require the screen to be scrolled sideways to reveal them all.

```
UID          PID  PPID  C    SZ   RSS PSR STIME TTY          TIME CMD
root           1     0  0 40044  7344   0 Nov15 ?        00:00:03 /sbin/ir
root           2     0  0     0     0   0 Nov15 ?        00:00:00 [kthreac
root           3     2  0     0     0   0 Nov15 ?        00:00:00 [rcu_gp]
root           4     2  0     0     0   0 Nov15 ?        00:00:00 [rcu_par
root           6     2  0     0     0   0 Nov15 ?        00:00:00 [kworker
root           8     2  0     0     0   0 Nov15 ?        00:00:00 [mm_perc
root           9     2  0     0     0   0 Nov15 ?        00:00:00 [ksoftir
root          10     2  0     0     0   0 Nov15 ?        00:00:02 [rcu_sch
root          11     2  0     0     0   0 Nov15 ?        00:00:00 [migrati
root          12     2  0     0     0   0 Nov15 ?        00:00:00 [idle_ir
root          14     2  0     0     0   0 Nov15 ?        00:00:00 [cpuhp/0
root          15     2  0     0     0   1 Nov15 ?        00:00:00 [cpuhp/1
root          16     2  0     0     0   1 Nov15 ?        00:00:00 [idle_ir
root          17     2  0     0     0   1 Nov15 ?        00:00:00 [migrati
root          18     2  0     0     0   1 Nov15 ?        00:00:00 [ksoftir
root          20     2  0     0     0   1 Nov15 ?        00:00:00 [kworker
root          21     2  0     0     0   1 Nov15 ?        00:00:00 [kdevtmp
root          22     2  0     0     0   0 Nov15 ?        00:00:00 [netns]
root          23     2  0     0     0   0 Nov15 ?        00:00:00 [rcu_tas
:
```

Pressing the "Right Arrow" key shifts the display to the left.

```
 PSR STIME TTY          TIME CMD
   1 Nov15 ?        00:00:03 /sbin/init splash
   0 Nov15 ?        00:00:00 [kthreadd]
   0 Nov15 ?        00:00:00 [rcu_gp]
   0 Nov15 ?        00:00:00 [rcu_par_gp]
   0 Nov15 ?        00:00:00 [kworker/0:0H-kb]
   0 Nov15 ?        00:00:00 [mm_percpu_wq]
   0 Nov15 ?        00:00:00 [ksoftirqd/0]
   1 Nov15 ?        00:00:03 [rcu_sched]
   0 Nov15 ?        00:00:00 [migration/0]
   0 Nov15 ?        00:00:00 [idle_inject/0]
   0 Nov15 ?        00:00:00 [cpuhp/0]
   1 Nov15 ?        00:00:00 [cpuhp/1]
   1 Nov15 ?        00:00:00 [idle_inject/1]
   1 Nov15 ?        00:00:00 [migration/1]
   1 Nov15 ?        00:00:00 [ksoftirqd/1]
   1 Nov15 ?        00:00:00 [kworker/1:0H-kb]
   1 Nov15 ?        00:00:00 [kdevtmpfs]
   0 Nov15 ?        00:00:00 [netns]
   0 Nov15 ?        00:00:00 [rcu_tasks_kthre]
:
```

The columns we now get are:

- **UID**: The user ID of the owner of this process.
- **PID**: The process ID of the process.
- **PPID**: Parent process ID of the process.
- **C**: The number of children the process has.
- **SZ**: Size in RAM pages of the process image.
- **RSS**: Resident set size. This is the non-swapped physical memory used by the process.
- **PSR**: The processor that the process is assigned to.
- **STIME**: Start time. The time when the process commenced.
- **TTY**: The name of the console that the user is logged in at.
- **TIME**: The amount of CPU processing time that the process has used.
- **CMD**: The name of the command that launched the process.

Listing Processes by Process ID

Once you have found the process ID for the process you're interested in, you can use it with the `ps` command to list the details of that process. Use the `-p` (select by process ID) option to achieve this:

```
ps -p 3403
```

```
dave@howtogeek:~$ ps -p 3403
```

The details for this process are listed:

```
dave@howtogeek:~$ ps -p 3403
  PID TTY          TIME CMD
 3403 tty2     00:00:08 shutter
dave@howtogeek:~$
```

You are not restricted to one process ID. You can provide a list of process IDs, separated by spaces.

Listing Processes by Command

The `-C` (command) option lets you search for a process using the command name. That is, the name of the command that launched the process. This is subtly different from the command line, which might include path names and parameters or options.


```
ps -C shutter
```

ps -C shutter in a terminal window

The details for the shutter process are listed.

Listing Processes Owned by a User

To see the processes that are owned by a particular user, use the `-u` (user list) option:

```
ps -u mary
```

ps -u mary in a terminal window

The processes owned by the user account mary are displayed.

Output from ps -u mary in a terminal window

Listing Processes by Terminal

To see the processes associated with a TTY, use the `-t` (select by TTY) option. Used without a TTY number, the `-t` option reports on processes associated with the current terminal window.

```
tty
```

```
ps -t
```

```
dave@howtogeek:~$ tty
/dev/pts/0
dave@howtogeek:~$ ps -t
  PID TTY      STAT   TIME COMMAND
 1320 pts/0    Ss     0:00 bash
 1334 pts/0    R+     0:00 ps -t
dave@howtogeek:~$
```

The `tty` command reports that this is pseudo-teletype 0. The processes listed by `ps -t` are all associated with TTY `pts/0`.

If we pass a TTY number on the command line, we should get a report of the processes associated with that TTY.

```
ps -t 1
```

This time the processes are all associated with TTY `pts/1`.

RELATED:

What Is A TTY On Linux? (And How To Use The Tty Command)

Selecting Columns to Display

With the `-o` (format) option you can select which columns you want to have included in the output from `ps`. You specify the columns by name. The (long) list of column names can be seen on the man page in the section titled "Standard Format Specifiers." In this example, we're choosing to have the CPU time (`pcpu`) and the command line with arguments (`args`) included in the output.

```
ps -e -o pcpu,args | less
```

```
dave@howtogeek:~$ ps -e -o pcpu,args | less
```

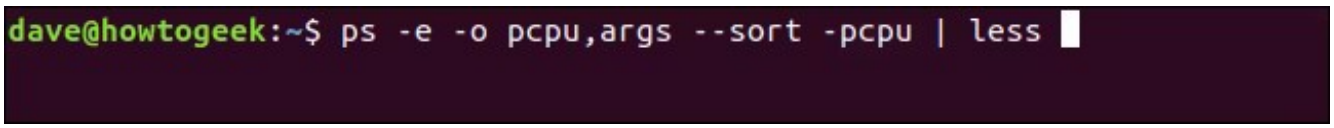
The output only includes our two requested columns.

```
%CPU COMMAND
 0.0 /sbin/init splash
 0.0 [kthreadd]
 0.0 [rcu_gp]
 0.0 [rcu_par_gp]
 0.0 [kworker/0:0H-kb]
 0.0 [mm_percpu_wq]
 0.0 [ksoftirqd/0]
 0.0 [rcu_sched]
 0.0 [migration/0]
 0.0 [idle_inject/0]
 0.0 [cpuhp/0]
 0.0 [cpuhp/1]
 0.0 [idle_inject/1]
 0.0 [migration/1]
 0.0 [ksoftirqd/1]
 0.0 [kworker/1:0H-kb]
 0.0 [kdevtmpfs]
 0.0 [netns]
 0.0 [rcu_tasks_kthre]
:
```

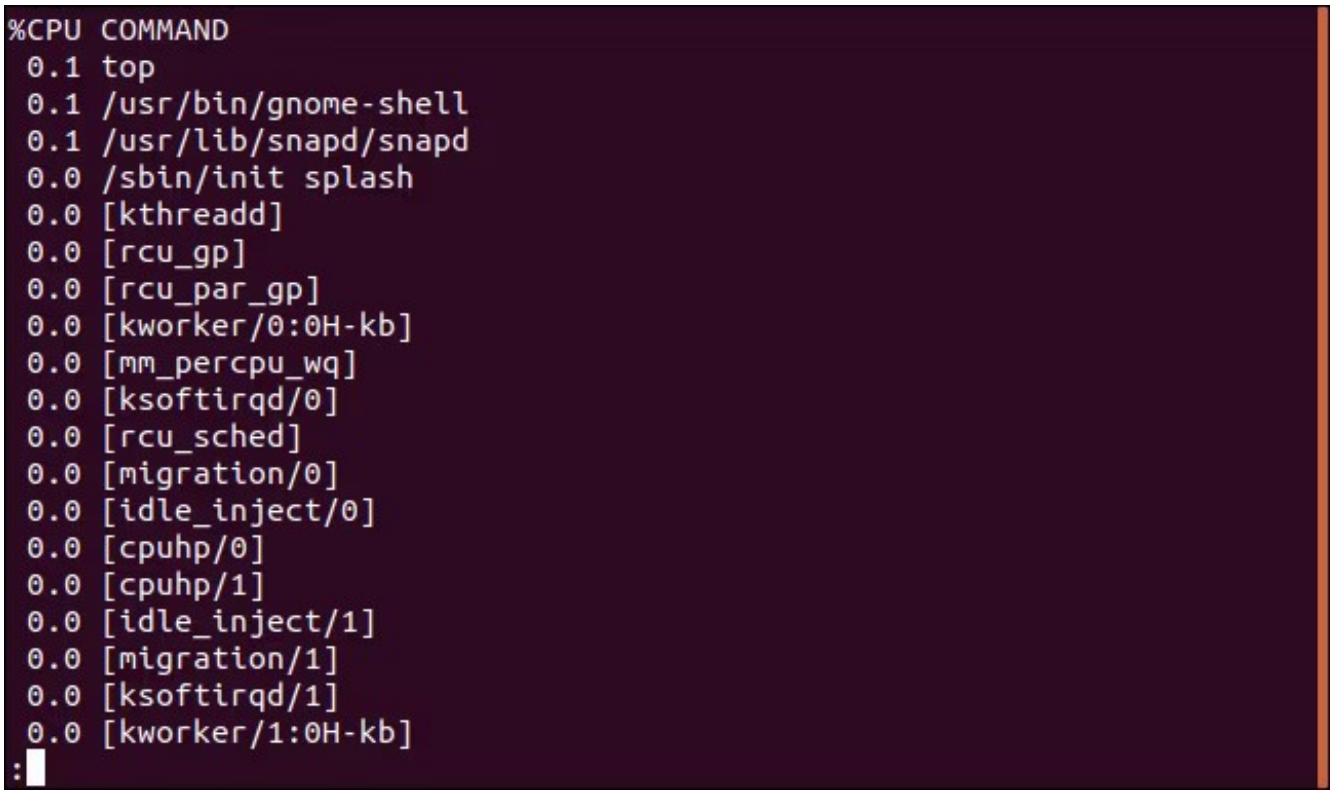

Sorting The Output by Columns

You can have the output sorted for you by using the `--sort` option. Let's sort the output by the CPU column:

```
ps -e -o pcpu,args --sort -pcpu| less
```

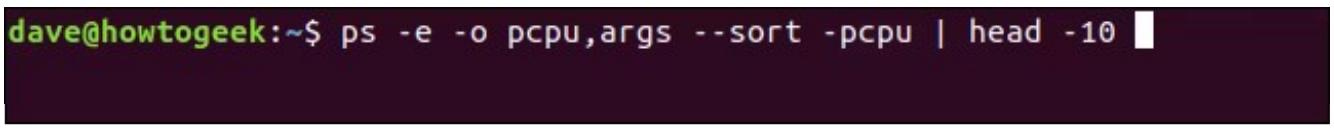


The hyphen "`-`" on the `pcpu` sort parameter gives a descending sort order.

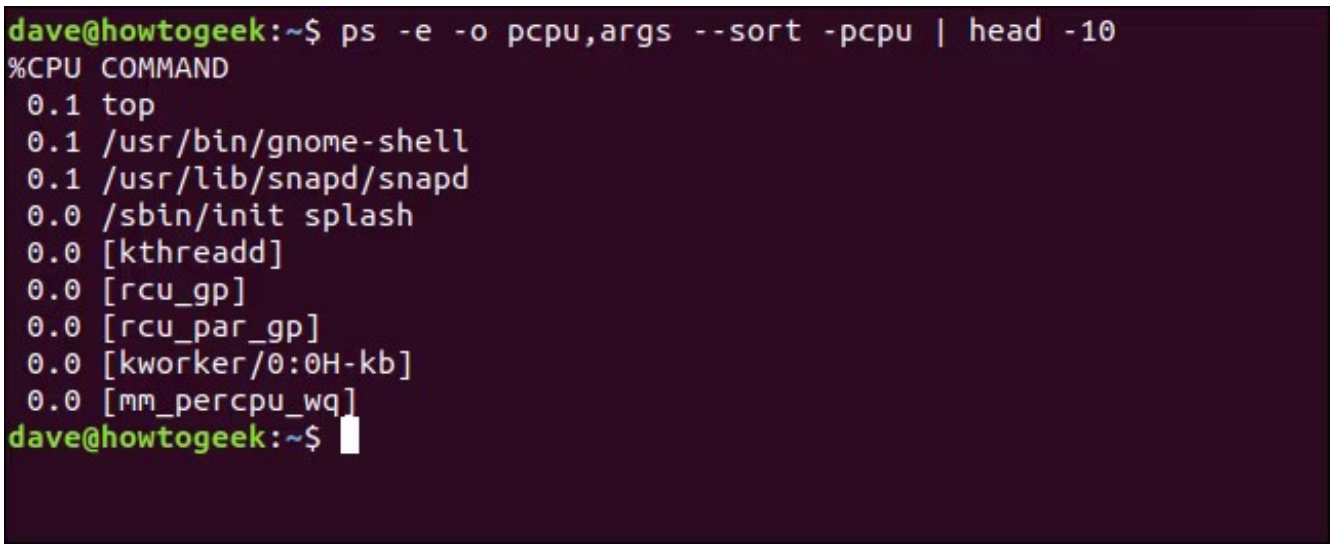


To see the ten most CPU intensive processes, pipe the output through the `head` command:

```
ps -e -o pcpu,args --sort -pcpu | head -10
```

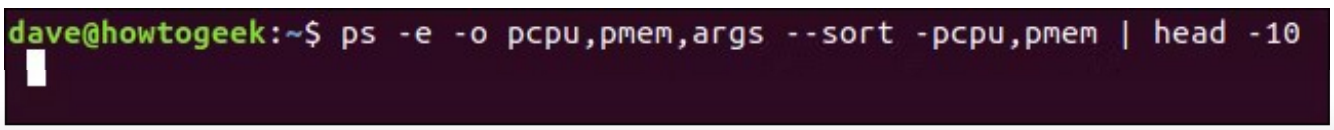


We get a sorted, truncated list.

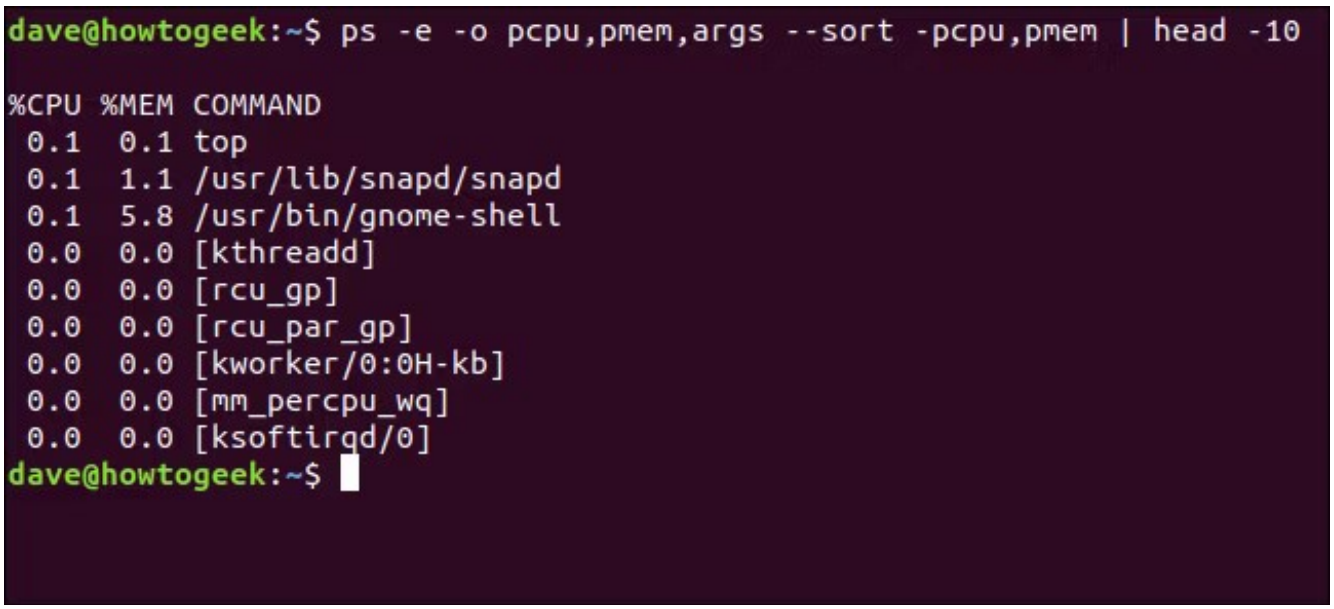


If we add more columns to our display, we can sort by more columns. Let's add the `pmem` column. This is the percentage of the computer's memory that is being used by the process. Without a hyphen, or with a plus " +", the sort order is ascending.

```
ps -e -o pcpu,pmem,args --sort -pcpu,pmem | head -10
```



We get our extra column, and the new column is included in the sorting. The first column is sorted before the second column, and the second column is sorted in ascending order because we didn't put a hyphen on `pmem`.



Let's make it a bit more useful and add in the process ID column (`pid`) so we can see the process number of each process in our listing.

```
ps -e -o pid,pcpu,pmem,args --sort -pcpu,pmem | head -10
```

```
dave@howtogeek:~$ ps -e -o pid,pcpu,pmem,args --sort -pcpu,pmem | head
 -10
```

Now we can identify the processes.

```
dave@howtogeek:~$ ps -e -o pid,pcpu,pmem,args --sort -pcpu,pmem | head
 -10
  PID %CPU %MEM COMMAND
  452  0.1  0.1 top
19892  0.1  1.1 /usr/lib/snapd/snapd
 1740  0.1  5.8 /usr/bin/gnome-shell
    2  0.0  0.0 [kthreadd]
    3  0.0  0.0 [rcu_gp]
    4  0.0  0.0 [rcu_par_gp]
    6  0.0  0.0 [kworker/0:0H-kb]
    8  0.0  0.0 [mm_percpu_wq]
    9  0.0  0.0 [ksoftirqd/0]
dave@howtogeek:~$
```


Killing Processes by Process ID

We've covered a range of ways to identify processes, including name, command, user, and terminal. We've also covered ways to identify processes by their dynamic attributes, such as CPU

usage and memory.

So, one way or another, we can identify the processes that are running. By knowing their process ID, we can (if we need to) kill any of those processes using the `kill` command. If we wanted to kill process 898, we'd use this format:

```
sudo kill 898
```

If all goes well, the process is silently terminated.

RELATED:

How To Kill Processes From The Linux Terminal

Killing Processes by Name

The `pkill` command allows you to kill processes by name. Make sure you've identified the correct process! This command will terminate the top process.

```
sudo pkill top
```

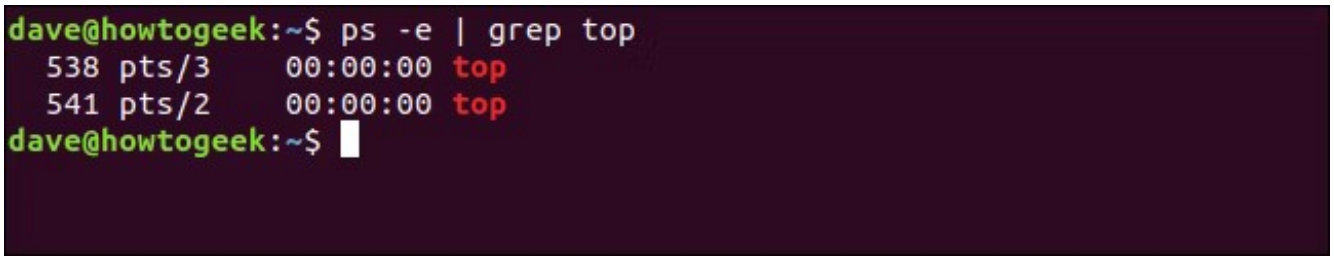
Again, no news is good news. The process is silently terminated.

Killing Multiple Processes by Name

If you have multiple copies of a process running, or a process has spawned a number of child processes (like Google Chrome can do), how can you kill them off? That's just as easy. We use the `killall` command.

We've got two copies of top running:

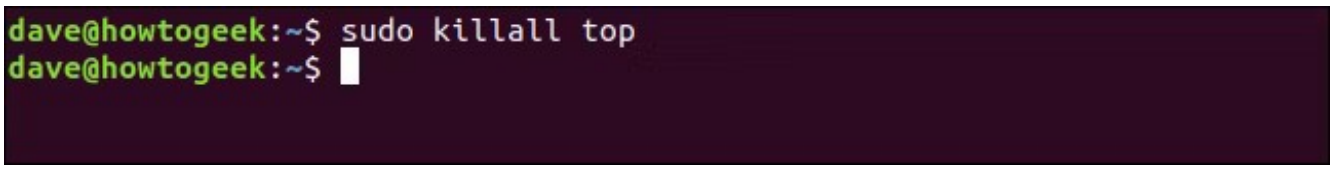
```
ps -e | grep top
```



We can terminate both of them with this command:

```
sudo killall top
```

No response means no problems, so both of those processes have been terminated.

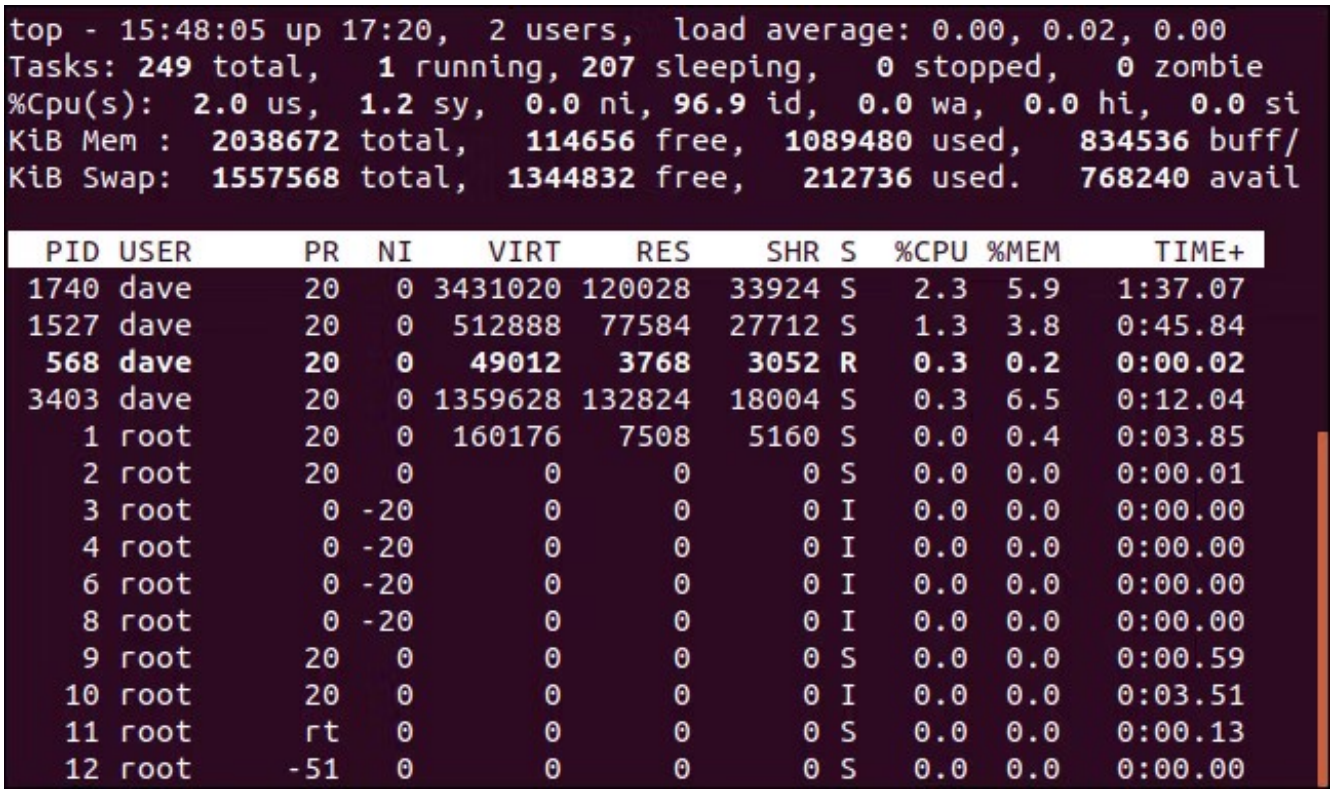


Get a Dynamic View with top

The output from `ps` is a snapshot view. It doesn't update. To get an updating view of the processes, use the `top` command. It provides a dynamic view of the processes running in your computer. The display is in two parts. There is a dashboard area at the top of the screen made up of lines of text, and a table in the lower part of the screen made up of columns.

Start `top` with this command:

```
top
```



The columns hold information on the processes:

- **PID**: Process ID
- **USER**: Name of the owner of the process

- **PR**: Process priority
- **NI**: The nice value of the process
- **VIRT**: Virtual memory used by the process
- **RES**: Resident memory used by the process
- **SHR**: Shared memory used by the process
- **S**: Status of the process. See the list below of the values this field can take
- **%CPU**: the share of CPU time used by the process since the last update
- **%MEM**: share of physical memory used
- **TIME+**: total CPU time used by the task in hundredths of a second
- **COMMAND**: command name or command line (name and command line parameters) If the command column cannot be seen, press the "Right Arrow" key.

The status of the process can be one of:

- **D**: Uninterruptible sleep
- **R**: Running
- **S**: Sleeping
- **T**: Traced (stopped)
- **Z**: Zombie

Press the "Q" key to exit from top.

RELATED:

37 Important Linux Commands You Should Know

Before You Kill a Process

Make sure it is the one you're after, and check that it isn't going to cause you any problems. In

Related Topics

Processes

Networking

About The Author

Dave McKay

(402 Articles Published)

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...

RELATED: ***Best Linux Laptops for Developers and Enthusiasts***

ANDROID IPHONE

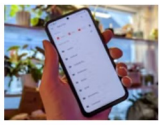
How to Change the Navigation Buttons or Gestures on Android
13 minutes ago

Is My Pixel Phone Waterproof?
4 hours ago

Moto Razr Is the Most Affordable Foldable in North America
20 hours ago

How to Transfer Contacts to a New Android Phone
1 day ago

How to Copy Music to Your Android Phone
1 day ago



How to Unzip Files on Android
4 days ago

See More

Home › Hardware

How Bad Are Power Outages for My Gadgets?

More power, more problems

BY SYDNEY BUTLER PUBLISHED 2 DAYS AGO

Hannah Stryker / How-To Geek

Readers like you help support How-To Geek. When you make a purchase using links on our site, we may earn an affiliate commission. ***Read More.***

KEY TAKEAWAYS

- **Power outages can cause data loss and corruption on electronic gadgets, especially if they occur during data writing or copying processes.**
- **Power surges can occur before or after a power outage, leading to an excessive voltage that can permanently damage electronic components. Power dips can pose a risk as well, especially if they are prolonged and low.**
- **Protect your gadgets by investing in surge protectors or uninterruptible power supplies, which ensure consistent power supply and protect against excessive voltage.**

No matter how reliable the power supply is where you live, there will inevitably be some point where you'll suffer an outage. With our homes now filled with sensitive (and expensive!) technology, how much harm can power outages do to our gadgets?

Power Outages Can Cause Data Loss

Most power events, be it a full power outage or a brown out, won't cause wall to wall chaos in your home with electronics going up left and right in a puff of smoke, thankfully. But because your electronic gadgets don't expect to be suddenly deprived of electricity, it stands to reason that anything they're working on at the point where the power goes out (or dips) can be interrupted or damaged. Maybe that work is something you're actively working on, maybe it's important stuff happening in the background on your router like an automatic firmware update.

This used to be a much bigger problem in the days when operating systems and critical files lived on mechanical hard drives, but with the growing dominance of solid state drives (SSDs) it's less of an issue.

However, it's still entirely possible that data on even modern SSDs can be corrupted if the power goes out. This is why your PlayStation 5 complains loudly if it wasn't switched off correctly. It's also why a power outage while a device is undergoing a critical update can cause, well, critical problems. The storage in the device might not be damaged, but the interruption of the firmware update can brick the device.

These issues are much more likely to happen if the power goes off while the device is copying or writing data. If it's only reading data, you'll probably be OK. Still, data loss remains the primary sort of damage power outages are responsible for, and although the damage may not be to your hardware, losing certain types of critical data is no less terrible for it.

Power Surges and Overvoltage Can Wreck Devices

You may not think too much power could be a side effect of having no power, but the truth is that power outages and power surges go hand-in-hand. There may be a surge in power either right before the power goes out, right after it comes back on, or both—which is why we highlight the dangers of the power surging when it comes back on in our guide to protecting your devices from power outages.

Voltage spikes (the "pressure" of the electrical current) are the main thing to worry about here, and if there's one thing electronics don't like, it's having too much voltage. If a component like a CPU or GPU gets too much voltage it can be permanently destroyed in an instant, and even devices such as power adapters and supplies that are designed to be somewhat robust against fluctuating power may eventually give up the ghost if you experience frequent surges.

Voltage Sags and Brownouts

There's a middle-ground between a power surge and a power outage, where the power dips lower than it should and then comes back. In most cases, these dips don't affect much because modern electronics have capacitors, voltage regulators, and other power supply components that help them deal with power dips that last a few milliseconds.

However, if the dips are too long and too low, it can be a lethal combination of power outage and power-on surge as the supply becomes erratic. This means you can have both data loss and voltage surge damage all in one go, and it can happen in a flash without you realizing anything has gone wrong. If the lights in your home or office are flickering or electronics are behaving erratically, but the power isn't completely out, you're experiencing volts sag, and your area might be experiencing brownouts.

How to Protect Your Gadgets From Poor Power Quality

Protecting your electronics from an erratic power grid isn't all that hard, but you want to plan in advance. The most effective protection is to invest in quality surge protectors. These devices will prevent from too much current or voltage reaching your electronics. The surge protector itself may be sacrificed in the process, but that's just a few dollars versus hundreds or thousands of dollars in damage.

Surge protectors won't protect you against data loss, though. The best solution for that is an uninterruptible power supply (UPS). These can be simple devices that only offer a few minutes of

power so you can safely turn off your gadgets. Then there are also modern UPS systems that also work as portable power stations that can run your equipment for hours if necessary. Most of these devices have surge protectors to protect themselves, and they ensure that your gadgets always get a consistent and clean power supply, no matter what happens at the mains. You can use a UPS for so much more than just a computer, too, as they work well for providing smooth and clean power to sensitive electronics of all kinds.

Using a laptop instead of a desktop computer is another obvious solution for computer use specifically since it essentially has its own UPS built in. However, you may still want to use a surge protector for the laptop's power brick to give it extra surge protection. The best thing about a laptop, though, is you can unplug it completely from the wall during a thunderstorm or other power event and keep working with zero risk a power surge will damage it. Speaking of unplugging things, if you don't have a UPS or surge protector, consider unplugging your devices until the power comes back on again. It usually only takes a second or two for restored power to stabilize, at which point you can plug them back in safely without worry about the device-wrecking surge that can come with a power grid restoration.

Related Topics

HARDWARE EXPLAINERS HARDWARE COMPUTER HARDWARE

About The Author

Sydney Butler

(436 Articles Published)

Sydney Butler is a technology writer with over 20 years of experience as a freelance PC technician and system builder. He's worked for more than a decade in user education. On How-To Geek, he focuses on creating commerce content with simpl...