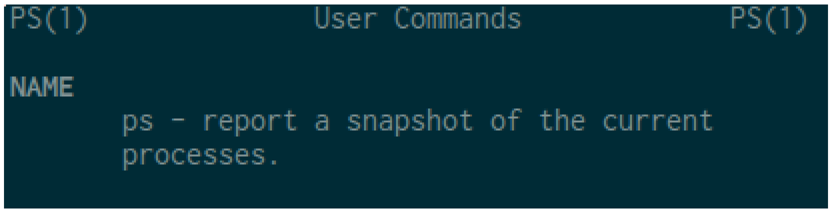
Last updated Saturday, Nov 16, 2019

Linux and Unix ps command tutorial with examples

Tutorial on using ps, a UNIX and Linux command for reporting information on running processes. Examples of searching by user, group, executable name and killing processes.

Estimated reading time: 5 minutes

Table of contents



```
PS(1)                 User Commands                PS(1)

NAME
       ps - report a snapshot of the current
       processes.
```

What is the ps command in UNIX?

The `ps` command reports information on current running processes, outputting to standard output. It is frequently used to find process identifier numbers. It supports searching for processes by user, group, process id or executable name. Related commands include `pgrep` that supports searching for processes and `pkill` that can kill processes based on a search.

How to show processes for the current shell

To show the processes for the current running shell run `ps`. If nothing else is running this will return information on the shell process being run and the `ps` command that is being run.

```
ps
PID  TTY          TIME CMD
5763 pts/3    00:00:00 zsh
8534 pts/3    00:00:00 ps
```

The result contains four columns of information.

- `PID` - the number of the process
- `TTY` - the name of the console that the user is logged into

- `TIME` - the amount of CPU in minutes and seconds that the process has been running
- `CMD` - the name of the command that launched the process

To demonstrate that other processes will show by just running `ps` a task can be put into the background before running the command.

```
sleep 10 &
ps
PID   TTY          TIME CMD
5763  pts/3    00:00:00 zsh
10254 pts/3    00:00:00 sleep
10258 pts/3    00:00:00 ps
```

How to list all processes

To list all processes on a system use the `-e` option.

```
ps -e
PID TTY          TIME CMD
  1 ?        00:00:01 systemd
  2 ?        00:00:00 kthreadd
  3 ?        00:00:00 ksoftirqd/0
....
```

This option can be combined with the `-f` and `-F` options to provide more information on processes. The `-f` option offers full-format listing.

```
ps -f
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 19:58 ?        00:00:01 /sbin/init
root         2     0  0 19:58 ?        00:00:00 [kthreadd]
root         3     2  0 19:58 ?        00:00:00 [ksoftirqd/0]
...
```

The `-F` provides extra full format information.

```
ps -F
UID        PID  PPID  C    SZ   RSS PSR STIME TTY          TIME CMD
root         1     0  0 13250  6460   1 19:58 ?        00:00:01 /sbin
root         2     0  0     0     0   1 19:58 ?        00:00:00 [kthr
root         3     2  0     0     0   0 19:58 ?        00:00:00 [ksof
...
```

Another commonly used syntax to achieve seeing every process on the system using BSD syntax is `ps aux`.

```
ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMM
root         1  0.0  0.0  53120  6368 ?        Ss   Sep19   0:01 /sbi
root         2  0.0  0.0      0     0 ?        S    Sep19   0:00 [kth
...
```

How to list all processes for a user

To list all processes by user use the `-u` option. This supports the user ID or name.

```
ps -u george
 PID TTY          TIME CMD
1053 ?        00:00:00 systemd
1062 ?        00:00:00 (sd-pam)
1074 tty1     00:00:00 zsh
...
```

How to list all processes for a group

To list all processes by group use the `-g` option. This supports the group ID or name.

```
ps -g users
 PID TTY          TIME CMD
 997 ?        00:00:00 login
1053 ?        00:00:00 systemd
1062 ?        00:00:00 (sd-pam)
...
```

How to list all processes by process number

To list all processes by process number use the `-p` option. This selects the processes whose numbers match the list provided to the `-p` option.

```
ps -p 12608 3995
  PID TTY      STAT   TIME COMMAND
 3995 ?        Ss     0:00 st
12608 pts/2    S+     0:04 vim content/post/unix-ps.md
```

How to list all processes by executable name

To list all processes by executable name use the `-C` option. This selects the processes whose executables match the list of executables given to the `-C` option.

```
ps -C tmux
 PID TTY          TIME CMD
5733 pts/0    00:00:00 tmux
5735 ?        00:00:06 tmux
```

How to show a process hierarchy or tree

To show a process hierarchy or tree use the `-H` option. This outputs a process tree.

```
ps -eH
  PID TTY          TIME CMD
 5735 ?        00:00:07   tmux
 5736 pts/2    00:00:00     zsh
12608 pts/2    00:00:08       vim
 5763 pts/3    00:00:00     zsh
17185 pts/3    00:00:00       ps
```

This may also be displayed in ASCII format by using the `--forest` option.

```
ps -e --forest
 PID TTY          TIME CMD
 5735 ?        00:00:07 tmux
 5736 pts/2    00:00:00  \_ zsh
12608 pts/2    00:00:08  |   \_ vim
```

```
 5763 pts/3    00:00:00  \_ zsh
16952 pts/3    00:00:00      \_ ps
```

How to just get the process id

A common task is to find the process id of a running process. Like many things in UNIX this can be achieved in a number of ways. In the examples above the `ps` command can look for processes by user, group or executable name. The `ps` can be also be piped to `grep` to search for arbitrary items.

```
ps -ef | grep vim
george   12608  5736  0 21:00 pts/2    00:00:11 vim content/post/unix
george   18324  5763  0 21:32 pts/3    00:00:00 grep vim
```

On many systems the `pgrep` command also exists that supports a number of ways to search for a process id. This is very useful if you are just interested in process id rather than other information. To search for all processes for an executable the `pgrep` command can be used.

```
pgrep tmux
5733
5735
```

To search by user pass the `-u` option.

```
pgrep -u george
1053
1062
1074
...
```

To search by group pass the `-G` option.

```
pgrep -G users
997
1053
1062
...
```

How to search for an kill a process

To search for an kill a process the `ps` command can first be used to find the process id before using the `kill` command to terminate the process. In the following example `ps` is piped to `grep` and `awk` .

```
sleep 100 &
[1] 21664
kill $(ps -e | grep 'sleep' | awk '{print $1}')
[1]  + terminated  sleep 100
```

On some systems the `pkill` command is also available to accomplish this.

```
sleep 100 &
pkill sleep
[1]  + terminated  sleep 100
```