10.2.7 Terminating Processes

Click one of the buttons to take you to that part of the video.

Terminate Processes 0:00-0:59

In this demonstration, we're going to talk about terminating processes. If possible, you should try to exit processes cleanly on your Linux system.

For example, if I'm running a user app like gedit, then to exit out of gedit, I should use the X button over here to clear it. Likewise, if I have a daemon running on the system, then I should use the systemctl command to shut down that service.

However, there may be times when these options don't work, and you have to actually kill a running process from the command line. For example, we may have a process running on the system and it's hung. No matter what we do using the traditional interface, we cannot get it to exit out.

In these cases, you can use several different commands to manually terminate that process from the command line. In this demo, we're going to look at these commands. The first one is kill. The second one is killall. The last one we're going to look at is pkill.

---

kill Command 1:00-1:35

Let's begin with the kill command. Let's go ahead and create a process that we can kill. I'm going to run 'gedit' again, but this time I'm going to run it in the background, just to make the shell prompt a little easier to work with.

We've got the gedit window running, which means there's a process now running for gedit. If we look right here, in the output of the command, we can see the process number of gedit. It's 3896.

The reason this is displayed here is because I did run gedit in the background. If you ran it in the foreground, then you'd have to open up a new terminal window and perhaps use the ps command to identify the PID that's been assigned to the process that you want to kill. Either way works.

---

kill Signals 1:35-4:54

The syntax for using the kill command is to enter 'kill', and then a dash (-), and then the kill signal that you want to send to the process, followed by the process ID--in this case, '3896'.

There are actually several different kill signals that I could specify right here. Let's go ahead and get rid of the PID, and let's just use the 'kill -l' command. -l stands for list, and it will display a list of all the different signals that we could potentially send the process, using the kill command.

Understand that some of these signals are more useful than others. There are four in particular that you should be familiar with. The first one is SIGHUP. This is kill signal number 1.

By the way, when you use the kill command, you can specify either the number associated with the signal or just the signal name itself--either one works. SIGHUP sends a signal to the process to restart. After the process has restarted, the process will be assigned exactly the same PID number that it had when it was running previously.

This is a really useful option if you've got a daemon running, and you've made several changes to the configuration file and you need to restart the service and you want it to have the same PID number that it had before, SIGHUP is one signal you can use.

Another signal you can use is SIGINT. This is signal number 2. This sends a Ctrl+c sequence to the process.

You can also send kill signal 9, SIGKILL. SIGKILL is more of a brute force process.

Understand that signals 1 and 2 may not kill a hung process. These signals work great for a process that's running and behaving properly. If the process is not behaving properly, you may have to go up to SIGKILL.

If you do this, be aware that the process may not be able to clean up after itself. In other words, the resources that were allocated to the process will remain allocated until you restart the system. In other words, it leaves a lot of junk behind.

Another option that you might want to look at is kill signal 15, SIGTERM. This signal tells the process to terminate immediately, but it does allow the process to clean up after itself before exiting, if it can.

SIGTERM is not nearly as brutal as SIGKILL. In fact, this is actually the default option. If you enter the kill command and don't specify a signal to send, it will send SIGTERM by default.

With that in mind, let's try sending a SIGTERM to our running gedit process. To do this, I enter 'kill' followed by a '-' and then either the name or the number of the signal I want to send. Let's send signal '15' to send a SIGTERM. Then the PID number of the process: '3896'.

Notice when I do that the gedit window closes. Because the process wasn't hung, it was running properly, it responded nicely to the SIGTERM signal.

Really, as a general rule of thumb, you should try the less brutal signals first. You should go to the meat cleaver, signal 9, only if nothing else works.

What I always recommend is to start with SIGINT; if that doesn't work, then try a SIGTERM; and then only if SIGTERM doesn't work should you go to SIGKILL because, as we said earlier, SIGKILL doesn't cleanly shut things down. It leaves a lot of junk floating around until you reboot the system.

---

killall Command 4:55-5:44

In addition to kill, you can also use the 'killall' command. The nice thing about killall is you don't have to know what the PID number is in order to kill a process. Let's go ahead and load gedit again so we have something to kill. Okay.

We've got gedit running now. It's running in the background. Notice that a new PID number has been assigned to the process, 3907. But we actually don't need that PID number.

With the killall command, we can send a particular signal. Let's do 'SIGTERM'. I'm sending exactly the same signal that I sent last time, just that last time I used a number; this time, I'm going to use the signal name.

It works exactly the same way. This time, because I'm using killall, I don't have to go and find out what the PID number is. I can just kill it by name, 'gedit'. Again, the gedit window disappears. Its process has been terminated.

---

pkill Command 5:45-7:04

In addition to kill and killall, you can also use a third command to stop a running process, and that's called pkill. pkill incorporates a lot of grep functionality with the kill command.

Basically, it allows you to search for and then kill running processes that match the search criteria that you specify. You can send those processes a particular kill signal.

Let's go ahead and load 'gedit' one more time. Load it again in the background. Let's suppose that I want to search through all of the running processes on the system and identify all of the running processes that have the text gedit in their name.

To do this, I would enter 'pkill' and then the signal that I want to send. Let's do a 'SIGTERM' again. Again, I could have used 15 instead of SIGTERM if I'd wanted to. It doesn't matter.

Then, I enter '-f' followed by the search term that I want to use. I'm going to use a very simple one this time, with just 'gedit', but I could also use regular expressions here.

By doing this, I could kill multiple processes at once instead of one at a time, which is kind of what you're stuck with if you use kill and killall.

For our purposes today, however, we'll just use it to kill one process, but do be aware that one of the key advantages of pkill is the fact that it can use regular expressions to kill lots of different processes all at once. Enter.

Again, the gedit window is gone. The process has been killed.

Summary 7:05-7:17

That's it for this demonstration. In this demo, we talked about how to kill processes, specifically how to kill hung processes. We first looked at using the kill command. We then looked at using the killall command, and then we ended this demonstration by looking at the pkill

command.

Copyright © 2022 TestOut Corporation All rights reserved.