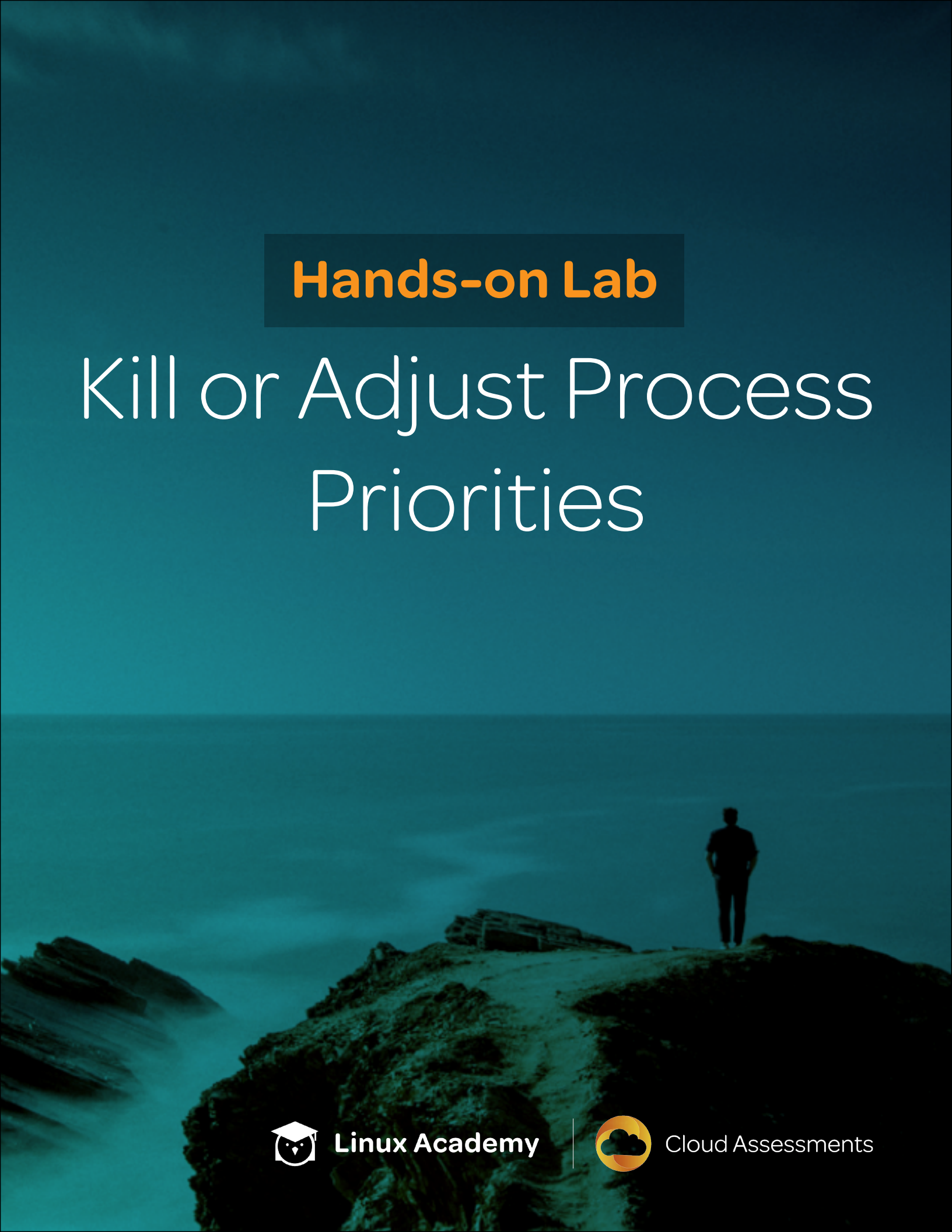
Hands-on Lab
Kill or Adjust Process Priorities
Linux Academy
Cloud Assessments

Contents

In this lab, we will go over how to review how much of your CPU is being used by a process, and then how to kill and adjust a process. To do so, we will be creating a program that will use a large amount of CPU. Please remember the password you create at the beginning of this lab! It will be needed near the end.

Checking CPU Consumption

Enter the command `top` to show the computer's base statistics. Our main focus for this lab will be with CPU. The CPU should generally be low, so we need to create a program that will cause a high influx to the CPU. To do that, we'll first use the `dd` command and manually create a large file to use in our script.

For the lab, we are using `/dev/urandom`, which randomly generates numbers, as the input file. We are creating a file called `testfile`. It will contain a count of 20 data blocks with each block being 1204 kilobytes.

```
[user@linuxacademy ~]# dd if =/dev/urandom of=testfile count=20
bs=1024k
```

To see how large the file is, use `du -sh testfile` to check the size of the testfile. Ours will be 20 MB in size.

With the file created, we now need a program to run that file. Using your chosen text editor, create a shell script. We will be naming it cputest.sh for this example.

```
[user@linuxacademy ~]# $EDITOR cputest.sh
```

You can either insert a short, looping shell program of your own creation or the following pre-made one. This script will constantly run, creating a high CPU scenario.

```
#!/bin/sh
i=0
while [ 1 ]
do
md5sum testfile
i=\`expr $1 + 1\`
echo “Iteration: $i”
done
```

Save the program and return to the main terminal.

To make the file an executable script, use the `chmod` command.

```
[user@linuxacademy ~]# chmod +x cputest.sh
```

Testing the High CPU

Open a second terminal window using **CTRL+T** and connect to the same server as tab 1. This way, you can view the `cputest` running and also use the `top` command view at the same time. You do not have to do this to complete the lab; it is only to help you see how your shell command program is responding to your kill commands.

In tab 1, run your `cputest` program.

```
[user@linuxacademy ~]# ./cputest.sh
```

Use the `top` command in tab 2. Here we can see that the CPU is much higher.

Killing a Command Process

There are a few different ways we can end a command and lower the CPU usage. The first is by using **CTRL+C**, which will appear as `^C` on the terminal, to kill the `./cputest.sh` script in our first tab. This command will end any process running in the terminal; in this case, `cputest.sh`.

```
[user@linuxacademy ~]# ./cputest.sh
[user@linuxacademy ~]# ^C
```

The other is to find the Process ID (PID) for the command in tab 2. Using `top`, search for `cputest.sh` under **COMMAND**. Execute a kill command for the PID, replacing `156241` with the PID used on your server. Note that each time you restart a process, the PID will change.

```
[user@linuxacademy ~]# kill 15624
```

Go back to the first tab. You will see that the program has stopped.

The last way we can kill the process is to use the `killall` command. This is especially useful if we have multiple of the same command running at once.

```
[user@linuxacademy ~]# killall cputest.sh
```

All versions of the program will end.

Changing a Process' Priority

Now say that instead of wanting to kill a process, we want to change its priority. Using the `nice`

command allows you to do just that. The default priority for a program is 20, with numbers higher than 20 being lower priorities, and numbers lower than 20 being higher priorities. For now, we are going to change the process to be a lower priority with the `nice` command in tab 1.

```
[user@linuxacademy ~]# nice -12 ./cputest.sh
```

Go to tab 2 and enter the `top` command. The -12 will change the `PR` for `cputest.sh` to 32, and the number 12 will appear in the `NI` column. Now, you can continue to change the priority for the process by using `renice`. For this example, we'll make the new priority 32. Remember, a larger number means a lesser priority. Note that the PID will have changed; this is because we ended the process before and are starting a new run of the process. Use the new PID for your process in the following command:

```
[user@linuxacademy ~]# renice 16 –p 10851
```

A noticeable difference between `nice` and `renice` is that in `nice` must have a `-` before the number to create a lesser priority. With `renice` we do not add `-` before the number to create a lower priority. In `renice` a `-` before the number will subtract it from the default NI number, 20, making it smaller instead of larger, and therefore raising the priority.

This time let's change the priority to higher than twenty. Now, normal users will not be able to complete this action. To allow the change in priority, use the `sudo` command in conjunction with `renice` and enter the `root` user password. This will be the same password you created when starting the lab.

```
[user@linuxacademy ~]# sudo renice -16 –p 10851
[sudo] password for user: $PASSWORD
18051 (process ID) old priority 16, new priority -16
```

Now the `NI` will appear as -16 and the `PR` number will be 4. To reset the priority back to 20, simply enter 0, making it so there is no value added or subtracted from the default priority.

```
[user@linuxacademy ~]# renice 0 –p 10851
```

Review

Having completed this lab, you now have the know-how to kill a process that is taking up too much CPU, or adjust the process' current priority. Congratulations, you now control which process shall live and die with a single stroke of the keyboard.