Server System Management - Linux

Lab 6 – Process management

# 

# Lab 6: Process management

1. Log in on your vm as your regular user, and type sudo adduser week6 to create a new user week6. Now put this user in the sudo group using sudo adduser week6 sudo.

sudo adduser week6

[set password - debian]

sudo usermod -aG sudo week6

**Note**

Try to run adduser without sudo. It won’t find the command. Why is that ? Check your path in your environment settings using the env command. Now compare that to when you run the env command with sudo.

You are mimicking the fact that you are root

1. Log in using user week6. Type groups to confirm you are a member of the sudo group (needed to use sudo to run commands as root user). Keep doing the rest of the lab as week6 unless explicitly told to do otherwise ! so don’t “become root” with su

su – week6

groups



3. Update your apt repository source index - there might be new or updated packages : good practice : ALWAYS apt update before installing a new package

sudo apt update

4. Let's upgrade all installed packages to their most recent version in Debian Buster; it's a good habit to do this from time to time. Don’t use dist-upgrade, just upgrade this time. It’s actually recommended to use upgrade except when doing major version upgrades.

sudo apt upgrade

5. Let's install the Apache webserver package apache2

sudo apt install apache2

After installation, Apache will already be running. However, for the sake of this lab (and also if you intend to run php scripts on it), you will need to configure it to use the prefork worker because we want to illustrate a server that forks a child process per handled http request.

Do the following things :

sudo a2dismod mpm\_event

sudo a2enmod mpm\_prefork

A screen shot of a computer program

Description automatically generated

**Note**

You may have noticed we chose the prefork version.

This is a version that will fork another process for each browser that connects to us at a single moment.

It's called prefork because in the config file, you can specify the maximum number of clients we are willing to serve simultaneously (to limit the stress on our system), and Apache will fork those in advance, so they are ready to serve requests at any time.

6. now restart the apache2 service using the systemctl command (apache is nice and tells you how) (QUIZ)

sudo systemctl restart apache2

**Note**

7. Let’s install the net-tools package

sudo apt install net-tools

8. Let's see if something is listening on port 80 using the netstat -plnt command

A screenshot of a computer

Description automatically generated

9. You can see something is running on port 80 ! but the PID/program name isn’t shown to your measely week6 user… how could you get it to show you the program name ?

sudo netstat -plnt

10. what program is listening on port 80 according to netstat ? (QUIZ)

Apache2

apache2

A screenshot of a computer

Description automatically generated

11. Try to get the homepage of your apache webserver using wget

wget <http://127.0.0.1>

cat index.html

A computer screen shot of a computer program

Description automatically generated

12. It was saved to a file index.html in the current directory. You can view it with less index.html

Find out where the source/default index.html welcome page is located by browsing through the one you downloaded (QUIZ)

/var/www/html/index.html

13. Let's use the text-based browser w3m to browse to our server (install it using sudo apt install w3m)

w3m http://127.0.0.1

A screenshot of a computer screen

Description automatically generated

14. \*crossover\* You should be able to connect to your webserver using your web browser on your windows 10/11 (HOST or VM from Windows Labs) by just entering its IP adres, i.e. 192.168.X.Y. , (use ip address show to find your address). **Keep that browser open, pointing to your webserver**. (SCREENSHOT)

A screenshot of a computer screen

Description automatically generated

15. Now launch the top utility. It will show you a list of processes running on your server. It will by default order them by how much CPU time they use: the hardest-working processes will be at the top.

The list will be refreshed every couple of seconds.

16. Now quickly press refresh in your browser a couple of times, still pointing to your webserver.

Immediately look at the top screen you still had running. You should see that one or more apache2 processes made it to the top of the list, because they have been working a bit harder, because they were serving the requests you just did.

17. How many processes are now running for Apache?

Leave the top & use ps aux to show all processes, including those outside our bash session. Then zoom in on apache using grep

ps -aux | grep apache2 | wc -l

9

18. Look at the first column of the output. It shows you the user associated with the process.

You should see that there is one apache2 process running as root! Why is that ?

Because of default configuration

19. As what user are the others running ?

The others as running as user www-data. Those are the children spawned (forked) by the initial root apache2, to be sitting there waiting for a request assigned to them. They drop their root privileges as soon as they start executing (shortly after the fork), because they do not need to listen to port 80 - they just receive an established http TCP connection from that apache2 parent that runs as root.

**Note**

They are actually all clones forked from the initial apache2 process that was launched by init (systemd) when the service was started (at boot, or, in our case, right after apt install)!

20. install the psmisc package using sudo apt install. Now type pstree to see a more graphical view of the processes running on your system.

Observe the root apache2 process; systemd is its parent. You will also see that this apache2 has children. Notice the number of children.

A screenshot of a computer program

Description automatically generated

21. Now type ps -ef |grep apache2; this is a slightly different view that will include the PPID (Process ID (PID) of the parent process).

You should now see the root apache2, with parent PID 1. You should see that the other apache2 processes have parent PID equal to the one apache2 that is running as root.

A screen shot of a computer

Description automatically generated

This makes the picture complete for us.

22. Let’s now learn about SIGNALS. Those are simple messages we can send to a running process using the kill command

**SIGCHLD**

• one of your child processes died and you need to reap the zombie

(so you know its exit status: 0 = success or something else = a (possibly minor) problem occurred)

**SIGINT**

• you are currently working but the user wants to interrupt whatever you are doing (for example, to go back to a prompt or menu) and they just pressed Ctrl-C

**SIGSEGV , SIGBUS**

• you had a segmentation fault (memory error); you should clean up, close open files, save some data and exit as soon as possible

**SIGHUP**

• you are still running, but the user just rudely closed their ssh window and there is no one reading what you write on the terminal anymore (hangup)

**SIGKILL**

• the kernel is forcibly killing you (no chance to override this)

**SIGTERM (terminate)**

• the system or user want you to stop, and instead of using your Exit command (or q key), they chose to send you a signal

**Note**

When shutting down or rebooting your server, systemd will eventually send all processes the SIGTERM signal to stop them.

For daemons (services running in the background) like apache, this is the only way to quit them, as there is no CLI or GUI visibly running; it's running in the background.

**Note 2**

A process may be malfunctioning or behaving badly, or it may prefer not to obey your request (yet, or at all) because it wants to save some data first, ...

If you really want it to stop and it is not responding to the SIGTERM signal for whatever reason, you can send the more drastic SIGKILL signal (point above) which will always work.

When rebooting or shutting down, systemd will first try SIGTERM, and if the process takes too long to obey, it will eventually kill it off using SIGKILL. That’s what you sometimes see when shutting down your vm.

**SIGQUIT**

• the user really wants you to stop and insists, but wants to be nice and give you a chance to shut down gracefully.

23. Many more signals exist (64 in total) in order to see a list of all possible signals (and their number), use kill (read the man page for more info)

A screen shot of a computer program

Description automatically generated

kill -l

24. The terminology for UNIX process management is always a bit morbid (daemons, killing, zombies, reaping, ...) and in this tradition, the program to send a signal is indeed called kill.

As a pleasant intermezzo, install frotz on your Debian using sudo apt install. Then copy the LURKING.DAT file from leho and get that file into your week6’s home directory.

A screen shot of a computer

Description automatically generated

Run frotz ~/LURKING.DAT

Now smell the hacker. QUIT when you’ve had enough.

25. By default, kill sends SIGTERM, but you can send any signal to a process using kill. We will now test some of these signals on our running Apache2 processes.

Let's look up the PIDs again (you may still have them on your screen or in your scrollback buffer, when you were using ps -ef or ps aux).

Let's use the pidof command this time to get apache2’s pid :

A screen shot of a computer

Description automatically generated

How many apache2 processes do you see? Should be the same number you got in question 18 & 20.

26. You remember one apache2 was running as root and the others were preforked to handle incoming requests, running with less privileges as user www-data.

You can simply grep for www-data in /etc/passwd and you will find it as a regular user account. That way you can also see the associated user id.

Try sudo su –login www-data Does this work ? (QUIZ)

No

No, the shell in /etc/passwd is defined to be/usr/sbin/nologin, (a program that just terminates and tells you you are not supposed to be working with this user account.)

27. When using ps -ef, you can see which apache process is the root parent.

Often (and also in our case, since we are not using our system heavily), the one with the smallest PID was launched first and will be the one running as root. This is the one that was launched by systemd. The others were forked by this root instance.

28. Try to send the (default) TERM signal to the root apache2 instance as user week6:

kill \*PID\* (Replace PID by the pid of your apache2 root instance). This should give you an error. Which one?

A screen shot of a computer

Description automatically generated

29. Why do you think you got an error?

Because the process Is running as root, and you are trying to kill it as a random user with no priviileges.

30. Let's try again with sudo. Root can send signals to any process, regardless of the user that is running it.

A screen shot of a computer

Description automatically generated

31. Use ps -ef, ps aux (possibly piped to a grep apache2 for convenience), or pidof, to see if apache2 is still running.

Normally you wil find that both the root parent (which you just sent SIGTERM to terminate it gracefully) and the children have died.

That's because, when asked to stop, the parent will in turn properly terminate its children.

32. Use the systemctl program intended to manage services, to start apache2 again

sudo systemctl restart apache2

**Note**

To just see a list of all services, running or not, just run systemctl without command-line arguments.

33. It MAY be possible that it is not running (when you check with pidof)!

That is because we terminated it without letting systemd handle things. It did shutdown gracefully, but some /run files were not removed. systemctl may refuse to start it. Or not.

34. Debian's Apache2 comes with a script called apache2ctl to properly (re)start the apache service. Use this to (re)start apache2:

sudo apache2ctl restart

35. Confirm with pidof or ps -ef that apache2 is running again.

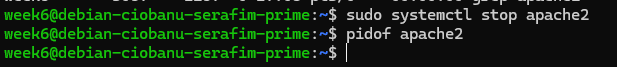
A screen shot of a computer

Description automatically generated

36. Shutdown and restart it using systemd (systemctl); this time there should certainly be no problems:

systemctl stop apache2

pidof apache2



# it should be gone

systemctl start apache2

pidof apache2

# it should be back now

A screen shot of a computer screen

Description automatically generated

What did you notice about the apache2’s pids when you restarted them ? (QUIZ)

They increased

37. Now try to sudo kill (SIGTERM) one of the apache2 children. Pick one of the higher pids.

A screen shot of a computer

Description automatically generated

How many apache2's are running now? (QUIZ)

6

The parent always detects that the child has died, because Linux sends parent processes the signal SIGCHLD automatically when one of their children stops or dies.

And it quickly made a new child

38.You could send any signal using kill. To make apache2 think it had a memory error, you could type this: sudo kill -SEGV \*pid of one of the child apache2s\*. Find the corresponding error in /var/log/apache2/error.log. (QUIZ)

A screenshot of a computer program

Description automatically generated

The error message in the log should indicate that a child process experienced a segmentation fault (SIGSEGV)

39. Learn about the pkill command. It is a convenience command, that allows you to specify the process by name rather than PID.

40. pkill apache2 will find all of your apache2 processes, whereas kill got just one PID from you.

sudo pkill apache2

41. Get the PID of a child apache2 process (that runs as user www-data). Restart apache2 if you killed it off before.

systemctl start apache2

pgrep -u www-data apache2

Explore the /proc filesystem. You will learn in OSC that (besides that everything is a file in Unix) you can talk to your kernel using this directory

Type this: (Replace pid by the PID of the child apache2 process you just obtained.)

cd /proc/\*pid\*

A screen shot of a computer program

Description automatically generated

42. You are now in a directory that is made up on the spot by the kernel (it is not a real directory on your disk!). Type ls

A screen shot of a computer screen

Description automatically generated

So, Linux gives you information about its processes by presenting you a fake directory tree for each process running on the system! Remember, in Unix/Linux, EVERYTHING IS A FILE

43. You will see a file called environ.

Display it using sudo cat. Read it; it contains interesting info.

A screen shot of a computer

Description automatically generated

Note

This is the environment for the process. These are the variables that are passed down to it by its parent

44. Find out the default Umask that the apache2 process uses. It's hidden somewhere in the process's proc folder's files

cat status

A screen shot of a computer

Description automatically generated

0022

45. That’s it for process management. You want to know how to configure our apache? That will be done in a more advanced course 😉

But to get started, just read the welcome page you've already opened in your browser. It’ll get you underway. Also, you can use the package manager to find the related config files.

sudo dpkg -L apache2|grep etc| less