Hands-on Projects

These projects should be completed in the order given. The hands-on projects presented in this chapter should take a total of three hours to complete. The requirements for this lab include:

* A computer with Fedora Linux installed according to Hands-on Project 2-1.

# Project 9-1

In this hands-on project, you view characteristics of processes using the ps command.

1. Boot your Fedora Linux virtual machine. Login to your chosen desktop environment as **user1** using password **LNXrocks!** and open a terminal window.
2. At the terminal, become **root** by typing **su -** and press enter and provide **LNXrocks!** as the password.
3. At the command prompt, type **ps -ef | less** and press Enter to view the first processes started on the entire Linux system.
4. Fill in the following information from the data displayed on the terminal screen after typing the command:
   1. Which process has a Process ID of 1?**/usr/lib/system/systemd**
   2. What character do most processes have in the terminal column (tty) and what does it mean? “**?”**
   3. Which user started most of these processes? **root**
   4. Most processes that are displayed on the screen are started by a certain parent process indicated in the Parent Process ID column (PPID). Which process is the parent to most processes? **kthreadd**
   5. Type q at the LESS prompt to quit.
5. At the command prompt, type **ps -el | less** and press Enter to view the process states for the first processes started on the entire Linux system.
6. Fill in the following information from the data displayed on the terminal screen after typing the command:
   1. What character exists in the State (S) column for most processes? **“S”**
   2. What range of numbers is it possible to have in the Nice (NI) column? **-20 to 0**
   3. Type q at the LESS prompt to quit.
7. At the command prompt, type **ps -el | grep Z** and press Enter to display zombie processes on your system.
8. **Provide screenshot(s) of steps 3 through 7.**

# Project 9-2

In this hands-on project, you use kill signals to terminate processes on your system.

1. Boot your Fedora Linux virtual machine. Login to your chosen desktop environment as **user1** using password **LNXrocks!** and open up a terminal window.
2. At the command prompt, type **ps -ef | grep bash** and press Enter to view the bash shells that are running in memory on your computer. *Record the* ***PID*** *of the* ***bash shell*** *running in your terminal as* ***user1****:* ***1897 1718; 2954 2946***
3. At the command prompt, type **kill -l** and press Enter to list the available kill signals that you can send to a process.
4. At the command prompt, type **kill -2 PID** (**where PID is the PID that you recorded in Question 2**) and press Enter. Did your shell terminate? **no**
5. At the command prompt, type **kill -3 PID** (**where PID is the PID that you recorded in Question 2**) and press Enter. Did your shell terminate? **yes**
6. At the command prompt, type **kill -15 PID** (**where PID is the PID that you recorded in Question 2**) and press Enter. Did your shell terminate? **no**
7. At the command prompt, type **kill -9 PID** (**where PID is the PID that you recorded in Question 2**) \***TAKE A SCREENSHOT NOW BEFORE PRESSING ENTER\*** and press Enter. Did your shell terminate? Why did this command work when the others did not?
8. **Provide screenshot(s) of steps 3 through 8.**

# Project 9-3

In this hands-on project, you run processes in the background, kill them using the kill and killall commands, and change their priorities using the nice and renice commands.

1. Boot your Fedora Linux virtual machine. Login to your chosen desktop environment as **user1** using password **LNXrocks!** and open up a terminal window.
2. At the terminal, become **root** by typing **su -** and press enter and provide **LNXrocks!** as the password.
3. At the command prompt, type **sleep 6000** and press Enter to start the sleep command, which waits 6000 seconds in the foreground. You will not get your prompt back until the 6000 seconds pass. To stop the command, send the process an INT signal by typing the **Ctrl+c** key combination.
4. At the command prompt, type **sleep 6000&** and press Enter to start the sleep command, which waits 6000 seconds in the background. Observe the background Job ID and PID that is returned.
5. Bring the background sleep process to the foreground by typing **fg %1** at the command prompt, then press Enter. Send the process an INT signal by typing the **Ctrl+c** key combination.
6. Place another sleep command in memory by typing the **sleep 6000&** command and pressing Enter. ***Repeat this command three more times to place a total of four sleep commands in memory*.**
7. At the command prompt, type **jobs** and press Enter to view the jobs running in the background. Recall what the + represents.
8. At the command prompt, type **kill %** and press Enter to terminate the most recent process and view the output.
9. At the command prompt, type **kill %1** and press Enter to terminate background job #1 and view the output.
10. At the command prompt, type **killall sleep** and press Enter to terminate the remaining sleep processes in memory. Verify that there are no more sleep processes in memory by typing the **jobs** command, then press Enter.
11. Place a sleep command in memory by typing **sleep 6000&** at a command prompt and pressing Enter.
12. Place a sleep command in memory with a lower priority by typing **nice –n 19 sleep 6000&** at a command prompt and pressing Enter.
13. Verify that these two processes have different nice values by typing the command **ps –el | grep sleep** at the command prompt and pressing Enter. ***Record the PID of the process with a nice value of 0*.**
14. At the command prompt, type **renice +10 PID** (*where PID is the PID you recorded in the previous question*) to change the priority of the process. Type the command **ps –el | grep sleep** and press Enter to verify the new priority.
15. **Provide screenshot(s) of steps 3 through 14.**

# Project 9-4

In this hands-on project, you view and manage processes using the top command-line utility.

1. Boot your Fedora Linux virtual machine. Login to your chosen desktop environment as **user1** using password **LNXrocks!** and open up a terminal window.
2. At the terminal, become **root** by typing **su -** and press enter and provide **LNXrocks!** as the password.
3. At the command prompt, type **top** and press Enter.
4. From the output on the terminal screen, record the following information:
   1. Number of processes: **261**
   2. Number of sleeping processes: **259**
   3. Amount of total memory (K): **6108328**
   4. Amount of total swap memory (K): **3145724**
5. While in the top utility, press the **h** key and observe the output. When finished, press the **q** key to return to the previous top output.
6. By observing the output under the COMMAND column on your terminal screen, identify the PID of the top command in the output and record it here: **2051**
7. Type **r** in the top utility to change the priority of a running process. When asked which process to change (renice), type the PID from the previous question. When asked which value to use, type **10** to lower the priority of the top process to 10.
8. Type **k** in the top utility to send a kill signal to a process. When asked which process, type the PID used in the previous question. When asked which signal to send, type 2 to send an INT signal.
9. At the command prompt, type **top** and press Enter.
10. By observing the output under the COMMAND column on your terminal screen, identify the PID of the top command in the output and record it here: **119**
11. Type **k** in the top utility to send a kill signal to a process. When asked which process, type the PID from the previous question. When asked which signal to send, type 15 to send a TERM signal. Did the TERM signal allow top to exit cleanly? **no**
12. **Provide screenshot(s) of steps 3 through 11.**

# Project 9-5

In this hands-on project, you schedule processes by using the at and crontab utilities.

1. Boot your Fedora Linux virtual machine. Login to your chosen desktop environment as **user1** using password **LNXrocks!** and open up a terminal window.
2. At the terminal, become **root** by typing **su -** and press enter and provide **LNXrocks!** as the password.
3. Schedule processes to run 1 minute in the future by typing the command **at now + 1 minute** at a command prompt, then press Enter.
4. When the **at>** prompt appears, type the word **date > /root/datefile** and press Enter.
5. When the second **at>** prompt appears, type the word **who >> /root/datefile** and press Enter.
6. When the third **at>** prompt appears, press the **Ctrl+d** key combination to finish the scheduling and observe the output.
7. In approximately one minute, type **cat datefile** and press Enter. View the output from your scheduled at job.
8. At the command prompt, type **crontab -l** and press Enter to list your cron table.
9. At the command prompt, type **crontab -e** and press Enter to edit a new cron table for the root user. If you receive an error regarding vi type the following at the command prompt and hit enter: **export EDITOR=/bin/vim** When the vi editor appears, add the line:

**30 20 \* \* 5 /bin/false**

1. When you finish typing, save and quit the vi editor and observe the output on the terminal screen.
2. At the command prompt, type **crontab -l** and press Enter to list your cron table. When will the /bin/false command run?
3. At the command prompt, type **cat /var/spool/cron/root** and press Enter to list your cron table from the cron directory. Is it the same as the output from the previous command?
4. At the command prompt, type **crontab -r** and press Enter to remove your cron table.
5. **Provide screenshot(s) of steps 3 through 13.**