OS Fundamentals   
Processes--Linux

Last Updated: 2/2/2021 11:21 AM

# Instructions

Answer all questions directly in this document. You will upload this completed document as your homework assignment.

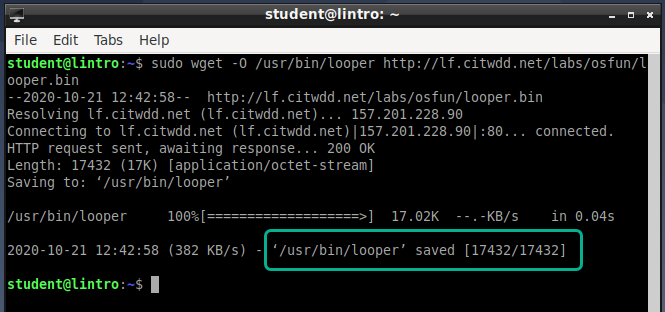
# Overview

This lab will explore Linux processes. You will create, kill, and change the priority or processes. You will also explore the /proc filesystem. You will use several terminal sessions throughout this lab.

# Setup

All you need is your Linux VM with an internet connection. You will need the looper.bin program to complete the lab. The looper.bin program is at

Login as student, start a terminal.

1. To install the looper.bin program enter the following command(s) from a **root** terminal (hint the –O is upper case).  
   sudo wget -O /usr/bin/looper <http://lf.citwdd.net/labs/osfun/looper.bin>  
   
2. now change the execute permission on the file.  
   sudo chmod +x /usr/bin/looper
   1. If you get an error about sudo you did not complete the access control lab properly.
3. You should get a message that the file was saved. If not try until it works.
4. By placing the file in the /usr/bin directory any user can run the program without specifying the program’s location.
5. Test the program by entering the command  
   looper
6. You should see a counter in the terminal, use the **Ctrl+C** key combination to exit the program.
7. Close all terminal sessions.

# Task 1—starting processes

## There are three primary ways to start processes. They are; via a terminal shell, via a graphical desktop manager, and via startup scripts. This task will focus on the first two.

## Steps

1. Start three terminals, I'll refer to them as Terminal 1 and Terminal 2, Terminal 3.
2. From **Terminal 1** 
   1. start the **looper** program.  
      looper
   2. You should see a counter just counting away.
3. From **Terminal 2** 
   1. change users to **bilbo**, remember you changed Bilbo’s password to **precious**.  
      su bilbo
   2. start the looper program  
      looper
4. Start the **galculator** program
   1. From the **Accessories** Menu select **Accessories🡪Galculator**
5. Start the **Image Viewer** program
   1. From the **Accessories** Menu select **Accessories🡪 Image Viewer**
6. You can minimize the galculator and Writer programs.
7. Now you will view the processes that are running, do not close any programs or terminals. From **Terminal 3**
   1. View the running processes with the ps command  
      ps  
      What processes are running? \_\_bash and ps\_\_
8. Background Processes. Sometimes you want to start a process in a terminal but then want to be able to run another command before the first one finishes.
   1. From **Terminal 1** (it should still be running looper)
      1. Can you enter any commands in the terminal? \_No, I can’t\_\_\_
   2. From **Terminal 3**
      1. Start the looper program as a background process.  
         looper quiet &
      2. You should see some output that shows the job number in square brackets and the process ID
      3. What is the job number? \_\_1\_\_  
         What is the Process ID? \_\_\_\_
9. Now what processes are running? From Terminal 3 enter the command  
   ps  
   What processes are listed? \_bash, looper, and psloop\_\_\_
10. Run the looper program as a background process in **Terminal 3** two more times  
    looper quiet &  
    looper quiet &
    1. Enter the ps command. Now what processes are shown? \_\_bash, looper, looper, looper, ps\_\_ How many instances of looper are running? \_\_3\_\_ Is that all of the looper processes on the system? \_No\_\_\_

# Task 2—the ps command

The ps command is used to show what processes are running. The ps command has several options, in this task you will explore some of them. When you look at the man page notice some options have a - in front of them and some do not. The **-a** is different than the **a** option.

## Steps

1. **From Terminal 3**
2. The **a** opton shows the process list not filtered by the current terminal user. Enter the command:   
   ps a
   1. How many processes are listed now? \_\_24\_\_ Hint you can use a command to figure this out. Remember wc will count lines. You could use the command ps a|wc – to get a count of the number of lines output. However that would list one more than is necessary because of the header information. You can fix that with the --no-headers option
      1. Enter the command   
         ps a --no-headers|wc -l
   2. How many instances of looper are running? Does this match your expectations? \_23\_\_
      1. You can use grep to help you out. Lets build the command  
         ps a |grep looper
      2. Now count it we don’t need the no headers option because grep filtered it out.  
         ps a|grep looper|wc -l
   3. The standard output has 4 columns.

|  |  |
| --- | --- |
| Column | Meaning |
| PID | Process ID |
| TTY | The terminal that started the process |
| TIME | How long the process has used the CPU |
| CMD | The command that started the process |

* 1. The o option allows you to specify which columns to show. Enter the command  
     ps ao tty --no-headers
  2. What TTY types are listed, enter the command? \_\_  
     ps ao tty --no-headers |sort|uniq
  3. List the terminal types ? \_pts/0, pts/1, pts/2, and tty1\_\_\_
  4. Notice the terminal types. The tty types are "physical" terminals they would include the physical keyboard and serial ports. The pts terminals are pseudo terminals and represent programs that create a software based terminal, like the terminals you are using.

1. The u option provides a more user-oriented output. Enter the command  
   ps u
2. The u option shows several additional columns of information.

|  |  |
| --- | --- |
| Column | Meaning |
| USER | Processes owner |
| PID | Process ID |
| %CPU | How much of the CPU the process is using |
| %MEM | How much memory the process is using |
| VSZ | Virtual memory used (in KB) |
| RSS | Real memory used (in KB) |
| TTY | The terminal that started the process |
| STAT | Status code |
| START | Time the process started |
| TIME | How long the process has used the CPU |
| CMD | The command that started the process |

1. The x option shows all processes even the ones that were not started by a terminal. Enter the command:  
   ps x
   1. Notice that many of the processes have a **?** in the terminal column. These processes were not initiated from a terminal. They include the kernel processes and daemon processes.
2. Finally, to see all processes in a more user friendly way combine the au and x options. Enter the command  
   ps aux
3. How many processes are running on your system? \_\_\_\_
4. Find the calculator (gcalctool) and Image Viewer (gpicview) processes and fill in the following table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | PID | Virtual Memory | Actual Memory | Status |
| galculator | 1196 | 364896 | 37388 | Sl |
| writer | 1204 | 59316 | 24208 | S |

# Task 3—signals

This task will explore sending signals to processes with the **kill**, **killall** and **top** programs.

## Steps

1. Terminal 1 should still have looper running, if not start it again.
2. **From terminal 3**
3. Since terminal 1 is the first terminal you started it should be using pts/0. Use the ps command to find the looper program that is running in pts/0  
   ps t pts/0  
   record the PID of the looper process \_\_1194\_\_
4. Review the signals  
   kill -l
   1. Record the signal number for the following signals

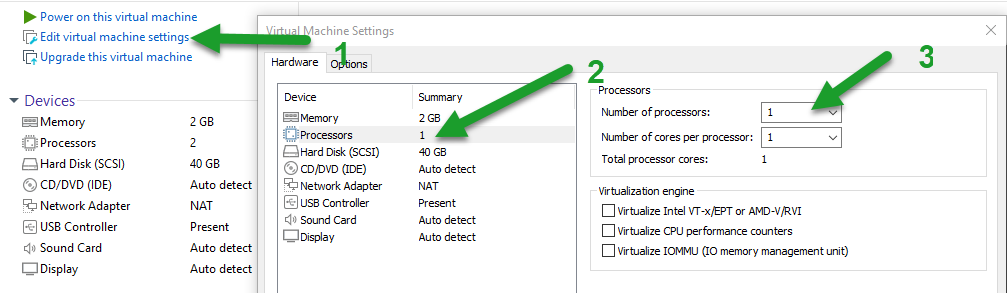
|  |  |  |
| --- | --- | --- |
| Signal (excluding the SIG prefix) | Signal number | Action |
| INT | 2 | Request to KILL, if not implemented kernel sends KILL, process should clean-up then KILL |
| STOP | 19 | Suspend execution |
| CONT | 18 | Resume a STOPed process |

1. Pause the looper process in terminal 1 using the kill command. The kill command takes two arguments the first is the signal name or number, the second is the process ID of the process to want to pause. Use the man page for exact syntax.
   1. Enter the appropriate command to suspend the looper process in terminal 1.
      1. Record the command used. \_\_Ctrl-z\_\_
   2. Now resume the program
      1. Record the command used. \_fg\_\_\_
   3. Finally stop the process
      1. Record the command used. \_\_script\_\_
2. Pause the process in terminal 2.
   1. **From Terminal 2** enter the keyboard command **Ctrl-z**
3. Discover the process ID of the stopped looper process.
   1. What is the process's PID? \_\_1195\_\_
   2. Record the commands you used here. \_\_Ctrl-z\_\_
4. **From terminal 2** restart the process using the approprate kill command
   1. Record the command you used here \_kill\_\_\_
   2. Pause the process gain using any method you would like.
      1. What method did you use? \_\_kill + the process ID\_\_
5. From terminal 3 kill the looper in terminal 2 with the kill command
   1. What was the result of the command? \_\_the process was stopped\_\_
6. Who is the owner of the looper process in terminal 2? \_root\_\_\_
7. What user is logged into terminal 3? \_\_root\_\_
8. You can solve the problem two ways, You can either switch to the root user or switch to Bilbo. For now, you will switch to Bilbo
   1. **From Termal 3** switch users to Bilbo  
      su biblo  
      Remember bilbo's password is precious
   2. Use the kill command to terminate bilbo's looper process in terminal 2. Record the command you used here? \_\_\_\_
   3. Exit the Bilbo shell. **From Terminal 3** enter the command   
      exit
9. Start several instances of looper in terminal 2. **From terminal 2** enter the commands  
   looper quiet &  
   looper quiet &  
   looper quiet &
10. From terminal 3 view the processes  
    ps au
    1. How many processes are owned by bilbo? \_\_\_
11. Now you will use killall to stop all processes owned by Bilbo. **From Terminal 3**
    1. Since you are logged in as student you will not be able to kill bilbo's processes as you discovered earlier. This time you will switch to a root shell. Enter the command   
       su
    2. Enter the command  
       killall -u bilbo
    3. Use the ps command to view processes. Does Bilbo have any processes running? \_\_\_\_

# Task 4—on being nice

This task will explore process priorities and the top program. The top program is a nice tool to get a quick look at what is happening on your system.

## Steps

1. Reconfigure your VM
   1. Shutdown your Linux virtual machine (don't suspend it shut it down).
   2. After the VM shuts down, edit the VM configuration and change the number of processors to 1.  
      
   3. Restart your VM
   4. Login as student
2. Start a terminal (this will be terminal 1)
   1. Switch to the user Bilbo  
      su Bilbo
   2. Start 3 background instances of looper and 1 foreground instance.  
      looper quiet &  
      looper quiet &  
      looper quiet &  
      looper
3. Start a second terminal (this will be terminal 2)
   1. Switch to root  
      su
4. **From terminal 2**
   1. Start the top program  
      top
   2. Top displays the most active processes on the system. You can see that our looper program is using most of the CPU resources of the system.
   3. Exploring top
      1. Press the ? key to get some help. Fill in the following table from the help.

|  |  |
| --- | --- |
| Function | Command |
| Quit |  |
| Set Update Interval |  |
| Show processes from specific user |  |
| Toggle color menu |  |
| Kill (send signal to process) |  |
| Change process priority (renice) |  |

* + 1. What is the update interval? \_\_\_\_
  1. Press esc key to exit help.
  2. See if you can find all the looper processes. There should be four of them.
     1. Record the PIDs here \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_ \_\_\_\_\_
  3. What is the approximate amount of CPU usage by all the looper processes combined? \_\_\_\_
  4. Are all the processes using approximately the same percentage of CPU time? \_\_\_\_

1. Viewing a process's priority. From top you can see a process's priority. The **NI** column shows the nice value. Nice values range from -20 to 19 with 0 normal priority. -20 has the highest priority 19 has the lowest or is the nicest.
   1. You can change a process's priority with the nice, renice or top programs.
   2. Open a new terminal and switch to root with the su command
   3. Make one of the process's more nice by changing its nice value to 10. Use the following command (replace the <pid> with the appropriate process ID number of one of the processes that is at the top of the list)  
      renice 10 <pid>
   4. After top refreshes describe the effect on CPU usage? \_\_\_\_
2. Close the last terminal you opened (terminal 3)
3. Now use the top program to change a process's priority.
   1. **From terminal 2** where top is running press the **r** key.
   2. Top prompts for a PID, enter the PID of one of the other looper processes.
   3. Change the nice value to **-10**
   4. Describe the results. \_\_\_\_
4. Viewing only certain user's processes
   1. Press the u key to only display processes from a specific user
   2. Enter bilbo
5. Sending signals with top
   1. You can send signals to processes with top. To send a signal press the **k** key
   2. Experiment sending signals to the looper processes. Make sure you leave at least one looper process running.
6. **From terminal 2** press **q** to quit top

# Task 5—exploring the /proc file system

Now you will explore the /proc filesystem to learn about the system and process configuration.

## Steps

1. From terminal 3 (you should still be root) get a directory of the /proc filesystem.  
   cd /proc  
   ls
2. Notice there a several directories that are just numbers. These directories represent processes running in the file system.
3. Now look at the file sizes  
   ls -l
4. What size are most of the files? \_\_\_\_
5. The proc file system is interesting since the kernel will only put data in the file when it is read. Most of the time most of the files will have a length of 0.
6. Explore the cpuinfo file  
   cat cpuinfo
   1. What is the model name of the CPU? \_\_\_\_\_
   2. What speed of processor do you have (cpu MHz)? \_\_\_\_\_
   3. What is the vendor\_id
7. Look at the memory configuration  
   cat meminfo
   1. What is the total amount of memory for your system? \_\_\_\_
   2. How much is free? \_\_\_
8. What version of Linux are you running (cat /proc/version)? \_\_\_\_
9. How busy is your system  
   cat /proc/loadavg
   1. The first three columns measure CPU utilization of the last 1, 5, and 10 minute periods.
   2. The fourth column shows the number of currently running processes and the total number of processes.
   3. The last column displays the last process ID used.
   4. How busy has your machine been for the last minute? \_\_\_\_
   5. How many processes are running on your machine? \_\_\_\_
10. Now look at a process
11. Use the ps command to find the PID of one of the looper processes (if you don't have any running start one in terminal 1. Record the PID ? \_\_\_\_
12. Change to the process's directory cd /proc/*<PID>*  where <PID> is the PID you discovered above.  
    cd /proc/*<PID>*
13. List the contents of the directory  
    ls -l
    1. Look for the file exe what is it? \_\_\_\_
    2. Look for the file cwd what is it? \_\_\_\_
14. Notice the environ file. This contains the environment variables for the process. List the contents  
    cat environ
15. Notice how all the information is jumbled together. This is because the file just contains an array of strings. Strings in Unix are null terminated so if you replace the nulls with a return you should be able to read the information better. Try this command.  
    cat environ|tr "\000" "\n"

# Wrap-up

Shut down Linux and change your processor count back to 4 or whatever value you would like.

# Deliverable

Upload this document with completed answers to canvas.