M6 Lab: Process Control

CITA 171: OPERATING SYSTEM USE & ADMIN

Marco Gonzalez
SUNY CANTON | 34 CORNELL DRIVE, CANTON, NEW YORK 13617

TABLE OF CONTENTS

1 TABLE OF FIGURES

2 PREPARATIONS

Start the CITA 171 VM and log in. Open a terminal window.

3 PROCESS

A **process** is an abstraction of program execution. A (computer) **program** is a list of instructions for a computer to perform tasks.

4 PROCESS INFORMATION

The **ps** command with the **-ef** or the **aux** option displays a snapshot of the current process execution state. See Figure 1. In this example, the first ten (10) lines of the ps command output are shown.

```
(04/24 19:28:36) cita171@cita171-vm: ~
$ ps -ef | head
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 18:51 ?        00:00:03 /sbin/init splash
root           2       0  0 18:51 ?        00:00:00 [kthreadd]
root           3       2  0 18:51 ?        00:00:00 [rcu_gp]
root           4       2  0 18:51 ?        00:00:00 [rcu_par_gp]
root           5       2  0 18:51 ?        00:00:00 [slub_flushwq]
root           6       2  0 18:51 ?        00:00:00 [netns]
root           8       2  0 18:51 ?        00:00:00 [kworker/0:0H-events_highpri]
root           9       2  0 18:51 ?        00:00:00 [kworker/u2:0-events_unbound]
root          10       2  0 18:51 ?        00:00:00 [mm_percpu_wq]
(04/24 19:32:13) cita171@cita171-vm: ~
$
```

Figure 1. Process Information Using the ps Command

Note the **UID**, **PID**, **PPID**, and **CMD** columns. The UID column shows the **process owner** user IDs. The process owner of a process is the user who can manage the process besides the system administrators. The PID column shows the process IDs. Each process is assigned a unique number that is used to identify the process. The PPID column shows the processes' **parent process** IDs. A process is created from another process that the created process becomes the **child process** of the process that created the process.

The child-parent relationship can be viewed using the **pstree** command. See Figure 2. In this example, the first ten (10) lines of the pstree command output are shown. Note that the number in the parentheses is the PID. PID 1 is always a process called **systemd**, which is the mother of all the other processes.

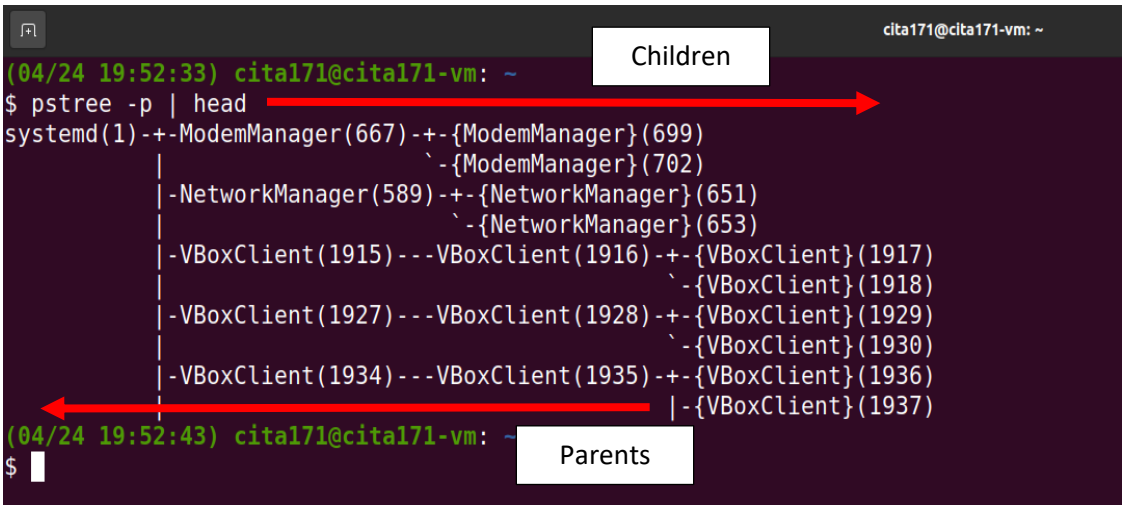

Figure 2. Child-Parent Process Relationship Using pstree Command

5 Real-Time Process Monitoring

The **top** command shows the current process activities. See Figure 3. Press the **Q** key to exit.

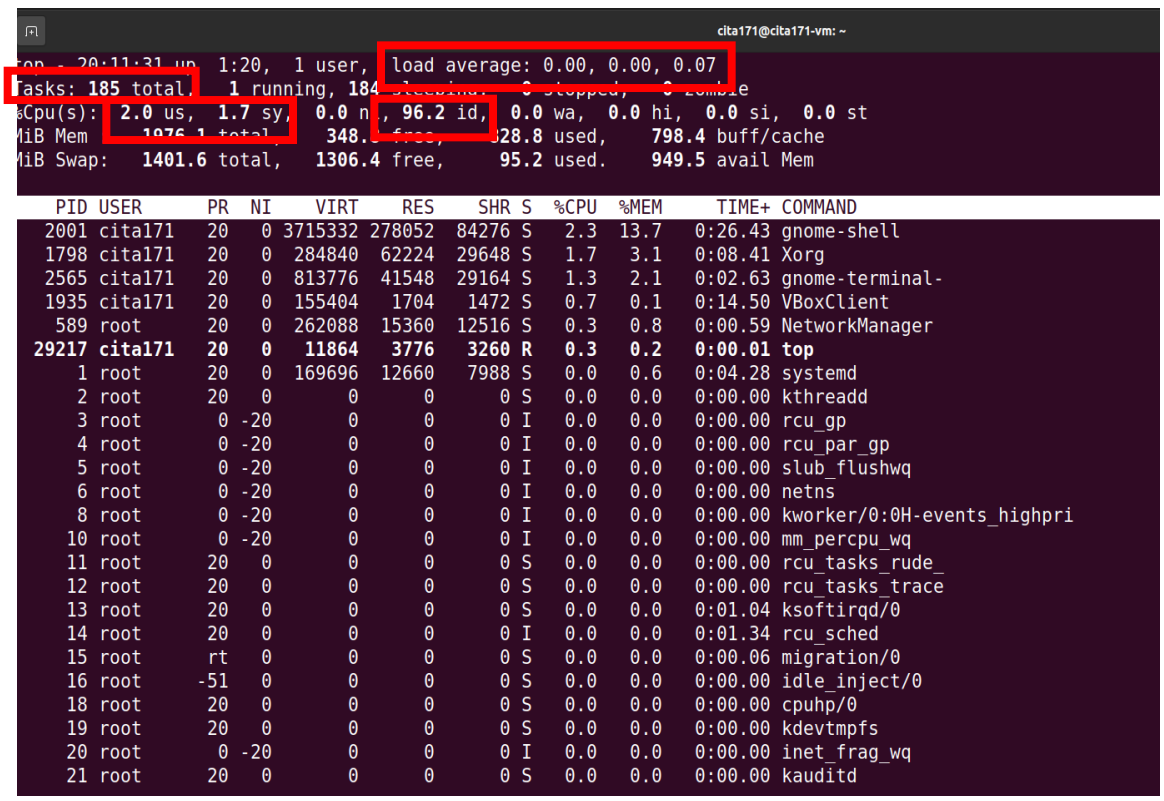

Figure 3. The top Command

This command displays process and memory information, including **Tasks** (the number of processes), **Load Average** (1 minute, 5 minutes, and 15 minutes averages), user (**us**) process load, system (**sy**) process load, idle (**id**) load percentages. Usually, the higher the id load is, the better.

6 Bash Job Management

6.1 Creating Bash Jobs

Bash jobs are processes that are created from a Bash shell process. See Figure 4. In this example, two nano processes are created by following these steps:

1. Execute **nano Message.txt** to start a nano process.
2. Type *This is Message.txt*.
3. Hold down the **Ctrl** key and press the **Z** key (**Ctrl+Z**). This keyboard shortcut stops the nano process.
4. Execute **nano Record.txt** to start a nano process.
5. Type *This is Record.txt*.
6. Press **Ctrl+Z**.

Figure 4. Starting and Stopping Two Bash Jobs

The numbers in square brackets are **Bash job IDs**. Like PIDs, Bash keeps track of its jobs by assigning each one a unique number.

6.2 Displaying Bash Jobs

The **jobs** command with the **-l** option shows the current Bash jobs and their corresponding Job IDs and PIDs. See Figure 5.

```
(04/24 20:35:10) cita171@cita171-vm: ~
$ jobs -l
[1]- 29240 Stopped (signal)        nano Message.txt
[2]+ 29242 Stopped (signal)        nano Record.txt
(04/24 20:35:15) cita171@cita171-vm: ~
```

Figure 5. The jobs Command

6.3 Switching Between Bash Jobs

The user can switch one Bash job to another job by job ID. Follow these steps to switch between Job IDs 1 and 2.

1. Type **%1** and press Enter. It resumes the **nano Message.txt** job (Job ID 1).
2. Press Ctrl+Z to stop the job.
3. Type **%2** and press Enter. It resumes the **nano Record.txt** job (Job ID 2).
4. Press Ctrl+Z to stop the job.
5. Type **%1** and press Enter. It resumes the **nano Message.txt** job (Job ID 1).
6. Press Ctrl+Z to stop the job.

6.4 Foreground and Background Processes

Foreground processes are the processes with which the user can interact directly. **Background processes** are the processes with which the user cannot interact directly. Follow these steps to test the differences. See also Figure 6 and Figure 7.

1. Type ping **127.0.0.1** and press Enter. A ping process starts as a foreground process. This process continues forever. In other words, the command prompt is not returned, no matter how long the user waits.
2. Press **Ctrl+C**. The ping process terminates. Because this ping process was a foreground process, the user was able to interact with the process directly to terminate it.
3. Type ping **127.0.0.1 &** and press Enter. (Note the ampersand character at the end.) A ping process starts as a background process.
4. Press **Ctrl+C**. The ping process does not terminate. Because this ping process is a background process, the user cannot directly interact with it to terminate it.
5. Close the Terminal program window.

```
(04/24 21:32:09) cita171@cita171-vm: ~
$ ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.028 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.079 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.080 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.085 ms
^C
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3074ms
rtt min/avg/max/mdev = 0.028/0.068/0.085/0.023 ms
```

Figure 6. Starting and Terminating a Foreground Process

```
(04/24 21:34:11) cita171@cita171-vm: ~
$ ping 127.0.0.1 &
[1] 29374
(04/24 21:34:15) cita171@cita171-vm: ~
$ PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.028 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.074 ms
^C
(04/24 21:34:16) cita171@cita171-vm: ~
$ 64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.043 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.044 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.068 ms
^C
(04/24 21:34:19) cita171@cita171-vm: ~
```

Figure 7. Starting and Failing to Terminate a Background Process

7 KILL COMMAND

The **kill** command is used to control processes. This command sends a special signal called a **kill signal** to processes, and they respond differently depending on the signal. When the command is used with the -l option, it lists 62 signals. See Figure 8.

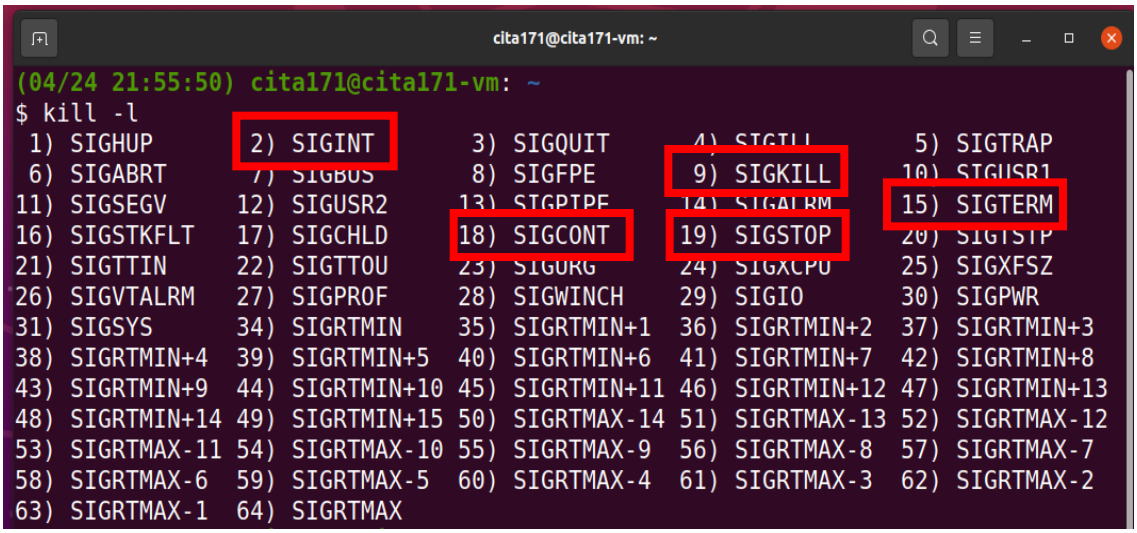

Figure 8. Kill Signals

By default, the kill command sends the **SIGTERM(15)** signal. This signal instructs the process to terminate gracefully. When the user presses Ctrl+C, the **SIGINT (2)** signal is sent to the process. When the user presses Ctrl+Z, the **SIGSTOP (19)** signal is sent.

In the following example, two Terminal program windows are laid out side-by-side. See Figure 9 and Figure 10. Follow these steps:

1. Type **ping 127.0.0.1** in one of the windows.
2. In the other windows, type **ps -el | grep ping** to get the PID of the ping process. (You may get a different PID. Use the one you got.)
3. Type **kill -SIGSTOP** followed by the ping PID and press Enter. The ping process stops. (Note that this command can also be executed as **kill -STOP** or **kill -19**.)
4. In the same window, type **kill -SIGCONT** followed by the ping PID and press Enter. The ping process resumes. (Note that this command can also be executed as **kill -CONT** or **kill -18.)**
5. In the same window, type **kill -SIGKILL** followed by the ping PID and press Enter. The ping process terminates. (Note that this command can also be executed as **kill -KILL** or **kill -9**. Under normal circumstances, the SIGKILL signal should only be used as the last resort. Use the default SIGTERM whenever possible.)
6. Click the ping windows and press Enter to confirm the process is killed.

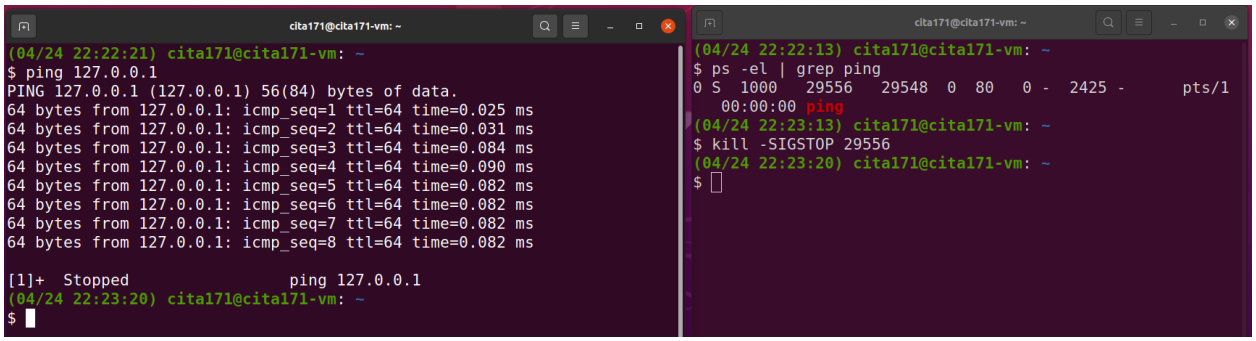

Figure 9. The SIGSTOP Signal

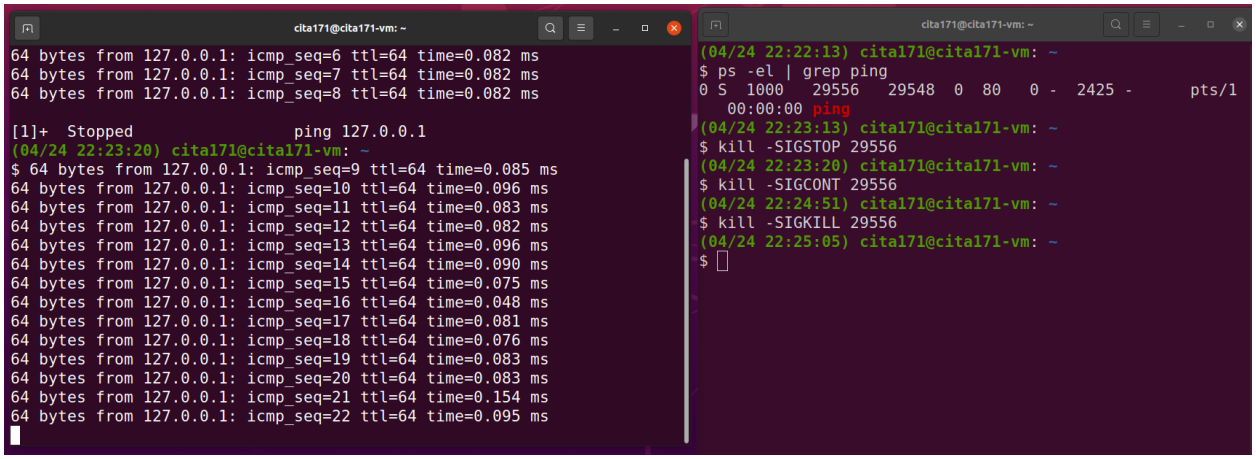

Figure 10. SIGCONT and SIGKILL Signals