OPS102 – Week 5 – Process Management - NDE/NDF

Introduction

Most of this lab requires you to preform steps, and then review the results. Most questions you simply answer to yourself, to confirm your understanding, and no explicit submission is required. Some steps require you to save a screenshot or other information and copy to the lab in blackboard. Once you have completed these labs steps, go to blackboard and answer the lab 5 questions there, and submit the lab in blackboard.

Both Linux and Windows, as powerful operating systems, provides robust process management capabilities. Understanding how to manage processes is crucial for effectively utilizing the operating system. A process refers to an executing program or task, whether it is a system service, a user application, or a background utility.

Here are some fundamental concepts related to process management:

- **Processes and Process IDs (PIDs):** Every process in Linux or Windows is assigned a unique identifier called a Process ID (PID). PIDs enable the system to track and manage processes effectively. You can view the PIDs of running processes using various commands and utilities.
- **Process States**: Processes can be in different states, such as running, sleeping, stopped, or terminated. Understanding these states helps in monitoring and controlling processes effectively. Commands like ps and top provide insights into process states.
- **Process Ownership**: Each process is associated with an owner, typically the user who initiated or owns the process. Process ownership is essential for managing permissions and access control.
- **Process Hierarchy**: processes follow a hierarchical structure. A process can create child processes, and those child processes can, in turn, spawn their own subprocesses. This hierarchical arrangement helps organize and manage related processes.
- **Process Control**: Linux provides various commands and tools to control processes. You can start, stop, pause, resume, or terminate processes using commands like kill, killall, pkill, and signals such as SIGSTOP and SIGCONT. Windows offers multiple methods to control processes. The Task Manager, a built-in Windows utility, allows you to view and manage running processes. It enables you to end processes, change process priorities, and analyze resource usage.
- F**oreground and Background Processes**: Both Linux and windows allow executing processes either in the foreground or background. Foreground processes run directly in the terminal, while background processes operate independently, freeing up the terminal for other tasks. You can switch between foreground and background using

commands like &, fg, and bg commands in Linux. In windows, Task Manager and PowerShell provide options to manage processes in both modes.

- **Process Monitoring and Resource Usage**: Monitoring the performance and resource usage of processes is essential for system administrators. In Linux, tools like top, htop, and ps provide real-time information on CPU usage, memory consumption, and other vital statistics. In Windows, Task Manager provides real-time information on CPU usage, memory consumption, disk activity, and network utilization. Performance Monitor (PerfMon) is a powerful tool for in-depth process monitoring.

Activity 1: Monitoring Linux Processes with ps command

Perform the following steps:

1. Make certain that you are logged into your Matrix account, and start the labtrack command: /home/john.sellens/tools/labtrack
2. Issue a Linux command to confirm that you are located in your **home** directory.
3. The **ps**_ command provides a list of processes that are running, or at least that were running at the time the command was called. Run the command `ps` in your terminal
4. ~~What output you see, take a screenshot and paste below.~~
5. How many processes are currently running? What information is displayed for each process? ~~Answer below.~~
6. Use the ps command with the '-e' option to display information about all processes in the system. Run the command **`ps -e`**
7. Analyze the output and identify the running processes on your system. Note the PID, TTY, and CMD columns. What do these column mean?
8. Use the 'ps' command with the '-f' option to display a full-format listing of the processes. Run the command **`ps -f`**
9. Examine the output, which provides detailed information about each process, including UID, PID, PPID, CPU%, MEM%, START, and CMD
10. Use the 'ps' command with the '-l' option to display a long listing format of processes. Execute the following command: **`ps -l`**
11. Analyze the output and observe the columns displayed, including F, S, UID, PID, PPID, PRI, NI, ADDR, SZ, RSS, WCHAN, STAT, TTY, TIME, and CMD.
12. Use the '**`-u`**' option followed by a username to display processes owned by that user.
13. Use the '**`-p`**' option followed by a process ID (PID) to display information about a specific process.
14. Type the exit command to exit out of labtrack.

Activity 2: Monitoring Linux Processes with top command

The `top` command is a powerful tool in Linux used to monitor and manage system resources in real-time. It provides a dynamic view of CPU usage, memory utilization, running processes, and other essential system metrics.

In this activity, sign in to matrix, and experiment with this command to understand resource usage. You might want to refer to the top(1) man page.

1. Run the command `top` in your terminal. What output do you observe, ~~below paste a screenshot of the terminal output~~?
2. Once the top command is running, you'll see a continuously updated display with various sections and columns.
3. Explain what information the following columns give.
   a. PR
   b. NI
   c. VIRT
   d. RES
   e. %CPU
   f. %MEM
   g. TIME+
4. The top command provides interactive features to customize the display and perform actions. Press 'P' to sort processes by CPU usage, 'M' to sort by memory usage, and 'N' to sort by PID.
5. To exit the top command, simply press 'q'. This will close the top display and return you to the terminal prompt.

Activity 3: Sending signals to processes

In Linux processes, system admins can send signals to communicate with processes and request specific actions. Signals are software interrupts delivered to a process by the operating system or another process. Signals allow processes to respond to various events, such as the termination of another process, user input, or changes in system conditions.

Signals are identified by unique numbers, known as signal numbers. Each signal number corresponds to a specific event or action.

Each signal has a default action associated with it, which determines what the process does when it receives that signal. Common default actions include termination, stopping, or ignoring the signal.

Common Signals: Linux systems have a set of standard signals defined, each with its own signal number. Some commonly used signals include:

- SIGTERM (Signal 15): This is the default signal sent by the kill command to request a process to terminate gracefully.
- SIGKILL (Signal 9): This signal immediately terminates a process. It cannot be caught or ignored.
- SIGSTOP (Signal 19): This signal pauses a process, suspending its execution until a SIGCONT signal is received.
- SIGCONT (Signal 18): This signal resumes the execution of a process that was previously stopped by a SIGSTOP signal.
- SIGHUP (Signal 1): This signal is typically sent to inform a process that the controlling terminal has been disconnected.

Signals can be sent to processes using the **`kill`** command.

Login to matrix, run labtrack (/home/john.sellens/tools/labtrack) and perform the following steps:

1. Issue the following command and observe that you must wait for the process (command) to finish after 10 seconds: **`sleep 10`**
2. Issue the following command: **`sleep 500 &`**
   The "sleep" command in Linux is a utility that allows you to pause the execution of a script or command for a specified amount of time. We will be using this command to simulate the behavior of a "long-running" process. This process will run in the background for **500 seconds**, and is not forcing the user to **wait** until this process finishes. A process that is **running in the terminal** is referred to as a **foreground process.** A processs that is running in the background does not block the user.
3. Run the command: **`ps`**
4. Note the process id of the background sleep command.
5. Run the command: jobs -l
6. Note that it also shows the same process id for the sleep command.
7. Run the command: **`kill PID`** (replace PID with process id)
   By default, the `kill` command sends the SIGTERM signal (signal number 15) to the process, requesting it to terminate gracefully. However, you can specify a different signal using the `-s` option followed by the signal number or signal name.
   What output you see? Paste a screensot of the output below.

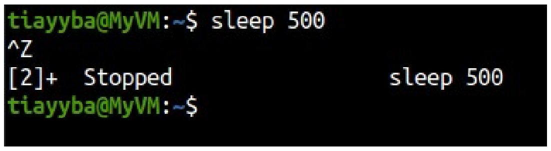
   Run the command “sleep 500 &” another time and this time send the SIGKILL singnal to this process. What output you see? Paste a screensot of the output below.
8. What difference you noticed in SIGTERM and SIGKILL singals?
9. Type the exit command to exit out of labtrack.

Activity 4: Foreground and background processes

1. Again ssue the following command:

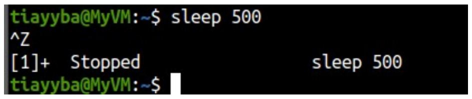
   `sleep 500`

   The Unix/Linux system is designed to allow users to send **preemptive signals** to manage those processes.
2. Press the following key combination to interrupt the process running on the terminal: **`ctrl-z`**.
   This sends a SIGSTOP signal to the process.
3. You should see output similar to what is displayed below:



```
tiayyba@MyVM:~$ sleep 500
^Z
[2]+  Stopped                 sleep 500
tiayyba@MyVM:~$
```

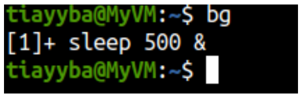
4. This indicates that this process has been placed into the **background**. This is useful in order to "**free-up**" the terminal to run other Linux commands
5. Issue the following Linux command: **`jobs`**
   You should see the following output:



```
tiayyba@MyVM:~$ sleep 500
^Z
[1]+  Stopped                 sleep 500
tiayyba@MyVM:~$
```

   This display indicates that this process (that is now in the background) has **stopped**.
   In other words, the *sleep* command is NOT counting-down to zero to terminate.

6. The plus sign "+" indicates the most recent process placed into the background.
7. Sometimes you would like to run the process you stopped in the background. You can use bg command without arguments to run in background the most recent process that was stopped.
8. Run the command: **`bg`**
9. Issue the command: **`jobs`**
10. You should see the following output similar to what was displayed above



```
tiayyba@MyVM:~$ bg
[1]+ sleep 500 &
tiayyba@MyVM:~$
```

11. The & sign indicates that the process is now running in the backlground.
12. You can also bring this process to foreground using fg command.
13. Issue the command **`fg`**. This will make the sleep process run in foreground.

Activity 5: Managing Windows Processes with PowerShell

Mostly Task Manager application is used for managing processes on Windows. However, Windows PowerShell does provide some commands for process management. The main command used to get information about process is called '`Get-Process`'. In the following tasks use this command in Windows PowerShell to get information about the process.

1. Run the `Get-Process` command in PowerShell and explain the output
2. Explain the meaning of column headers of the information output by this command
3. To get information about specific process you can use the syntax `Get-Process <process-name>`. For example, to get information about firefox process you can run command `Get-Process firefox`. Run this command to get information about a process of your choosing and take and save a screenshot for later submission.
4. Using this same command describe with example how you can get information about multiple processes
5. To stop a process you can use `Stop-Process` command with syntax `Stop-Process <process-name>`. In this task use this command to stop some process and show the screenshot below
6. There are two other commands of this class to manage processes. These commands are `Wait-Process` and `Debug-Process`. Search and read about these commands and provide examples.

Further Practice Questions.

Answer the following questions based on your knowledge of process management in Linux. You do not need to submit these answers – these questions are to reinforce your understanding of the material.

1. What is a process in Linux? Answer:
2. Name three different states a process can be in, and briefly describe each state.

   a) State 1: Description:

   b) State 2: Description:

   c) State 3: Description:

3. Which command is used to list processes in Linux? Provide an example of its usage.
   Command:

   Example:

4. Explain the meaning of the following columns displayed by the ps command:

   a) PID:

   b) CPU%:

   c) MEM%:

5. How can you terminate a process in Linux? Describe two different methods.

   Method 1:

   Method 2:

6. What is the purpose of the top command in Linux? How can you sort processes using top?

   Purpose of top:

   Sorting processes in top:

7. Why is it important to exercise caution when terminating processes in Linux? Explain briefly.
8. Briefly explain the difference between the kill and killall commands in Linux.
9. True or False: Terminating a process with SIGKILL allows it to perform cleanup operations before termination. Answer:
10. Name two signals that can be sent to a process using the kill command, and briefly describe their effects.

    Signal 1: Effect:

    Signal 2: Effect:

And Finally

Remember to submit lab 5 in blackboard.