**OPS102 – Week 5 – Process Management - Sample Lab**

Student Name: Abcedi Ilacas

Student ID: 138180211

**Introduction**

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.

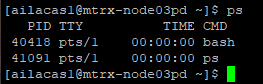
Here are some fundamental concepts related to process management:

* **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
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.



The ps command will show a snapshot of the current processes. By default, it displays processes for the current terminal session.

1. How many processes are currently running? What information is displayed for each process? Answer below.

There are 2 processes running. The information displayed includes the process ID (PID), terminal identifier (TTY), process time (TIME), and the command name (CMD).

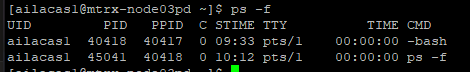
1. Use the ps command with the ‘-e’ option to display information about all processes in the system. Run the command **ps -e**
2. Analyze the output and identify the running processes on your system. Note the PID, TTY, and CMD columns. What do these column mean?

PID : This column displays the unique identifier assigned to each process running on the system. The process ID is a numerical value that allows the system and users to track and manage processes individually.

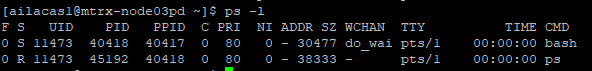
TTY : The TTY column shows the terminal associated with each process. If a process is associated with a terminal, this column will show the terminal identifier. Some processes that are not attached to a terminal will display a question mark (?) or might be blank in this column.

CMD : This column shows the command that initiated the process. It provides a quick view of the executable or script that is running as a particular process.

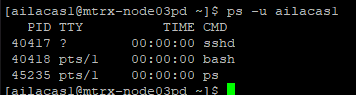
1. Use the 'ps' command with the '-f' option to display a full-format listing of the processes. Run the command **ps -f**
2. Examine the output, which provides detailed information about each process, including UID, PID, PPID, CPU%, MEM%, START, and CMD



1. Use the 'ps' command with the '-l' option to display a long listing format of processes. Execute the following command: **ps -l**
2. 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.



1. Use the '**-u**' option followed by a username to display processes owned by that user.



1. Use the '**-p**' option followed by a process ID (PID) to display information about a specific process.

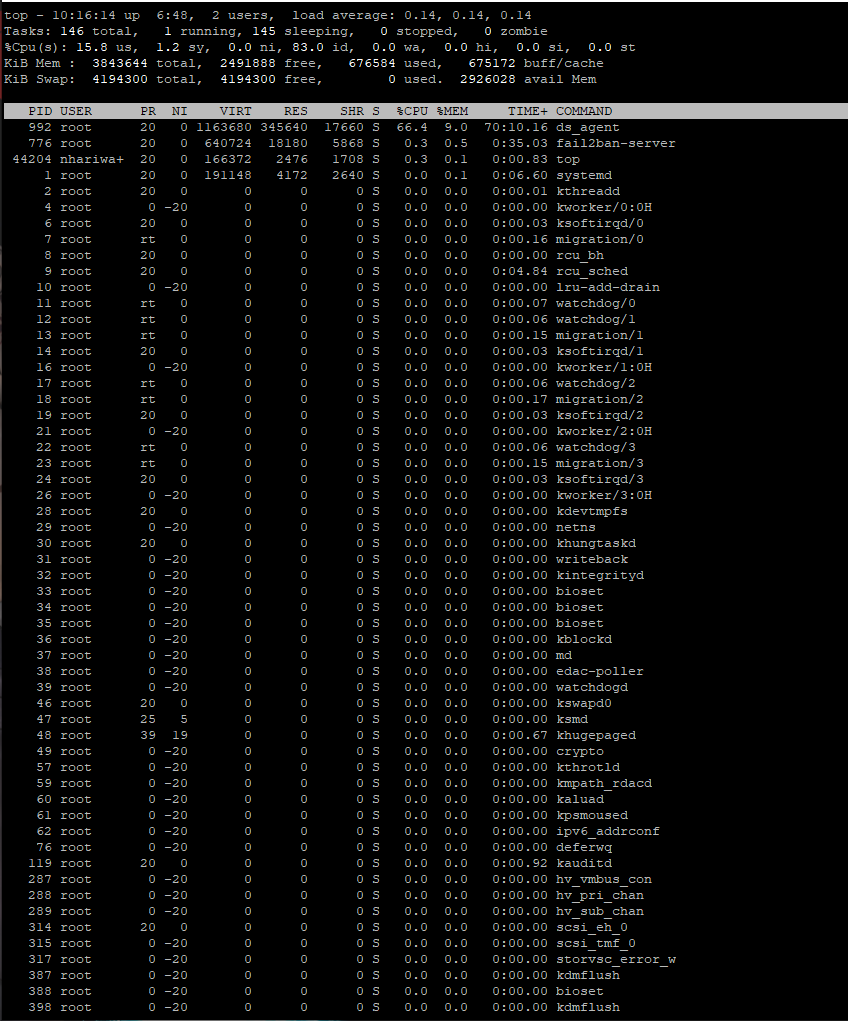


**Activity 2: Monitoring Linux Processes with top command**

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, experiment with this command to understand resource usage.

1. Run the command **top** in your terminal. What output do you observe, below paste a screenshot of the terminal output?



1. Once the top command is running, you'll see a continuously updated display with various sections and columns.
2. Explain what information the following columns give.
   1. PR - This column displays the priority of the processes. Lower numbers indicate higher priority. The kernel uses this value to determine which process should be executed next.
   2. NI - The nice value is a user-space setting that allows changing the scheduling priority of a process. Values range from -20 (highest priority) to 19 (lowest priority). A lower nice value means higher priority, while a higher nice value means lower priority.
   3. VIRT - This column shows the total amount of virtual memory used by each process, including all code, data, and shared libraries, plus pages that have been swapped out. The value is expressed in kilobytes (KB) or megabytes (MB).
   4. RES - This column displays the amount of physical memory (RAM) currently being used by each process, excluding the memory that has been swapped out. Like VIRT, the value is often expressed in kilobytes (KB) or megabytes (MB).
   5. %CPU - This column indicates the percentage of CPU time the process has consumed since the last update. It provides a measure of how processor-intensive a process is.
   6. %MEM - This column shows the percentage of total physical memory (RAM) being used by each process.
   7. TIME+ - This column displays the total amount of CPU time the process has used since it started. The time is reported in minutes:seconds or hours:minutes:seconds format.
3. 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.
4. To exit the top command, simply press 'q'. This will close the top display and return you to the terminal prompt.q

**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.

Perform the following steps:

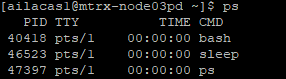
1. Issue the following command: **sleep 500**

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 for **500 seconds**, and is forcing the user to **wait** until this process finishes. A process that is **running in the terminal** is referred to as a **foreground process**.



I cant do anything now

1. Run the command: **ps**
2. Note the process id of sleep command.

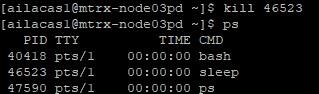


1. Run the command: **kill PID** (replace PID with process id)

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 screenshot of the output below.

No Output for this command



Run the command sleep 500 another time and this time send the SIGKILL signal to this process. What output you see? Paste a screenshot of the output below.

This command showed the PID, TTY, STAT, TIME and COMMAND of the sleep 500 command.



1. What difference you noticed in SIGTERM and SIGKILL signals?

The SIGTERM signal displayed nothing while the SIGKILL signal showed the PID, TTY, STAT, TIME and COMMAND of the sleep 500 command.

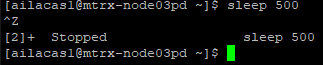
**Activity 4: Foreground and background processes**

1. Again ssue the following command:

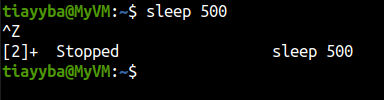
sleep 500

The Unix/Linux system is designed to allow users to send **preemptive signals** to manage those processes.

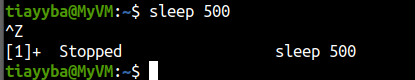
1. Press the following key combination to interrupt the process running on the terminal: **ctrl-z.** This sends a SIGSTOP signal to the process.



1. You should see output similar to what is displayed below:



1. 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
2. Issue the following Linux command: **jobs**  
   **You should see the following output:**



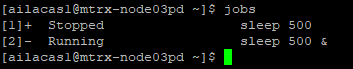


This display indicates that this process (that is now in the background) has **stopped**.  
In other words, the *sleep* command is NOT counting-down to zero to terminate.

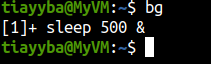
1. The plus sign "+" indicates the most recent process placed into the background.
2. Sometimes you would like to run the process you stopped in the background. You can use bg command without arguments to run in background the most recent process that was stopped.
3. Run the command: **bg**



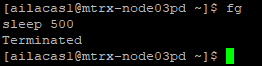
1. Issue the command: **jobs**



1. You should see the following output similar to what was displayed above



1. The & sign indicates that the process is now running in the backlground.
2. You can also bring this process to foreground using fg command.
3. 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



The output of Get-Process is similar to the top command we used in the previous activities in linux.

1. Explain the meaning of column headers of the information output by this command

Handles: The number of handles that the process has opened. A handle is a reference to a system resource, such as a file or a section of shared memory.

NPM(K): The amount of non-paged memory that the process is using, in kilobytes. Non-paged memory is memory that cannot be swapped out to disk and must remain in physical memory.

PM(K): The amount of pageable memory that the process is using, in kilobytes. Pageable memory is memory that can be swapped out to disk if necessary.

WS(K): The size of the working set of the process, in kilobytes. The working set is the set of memory pages currently loaded in physical memory.

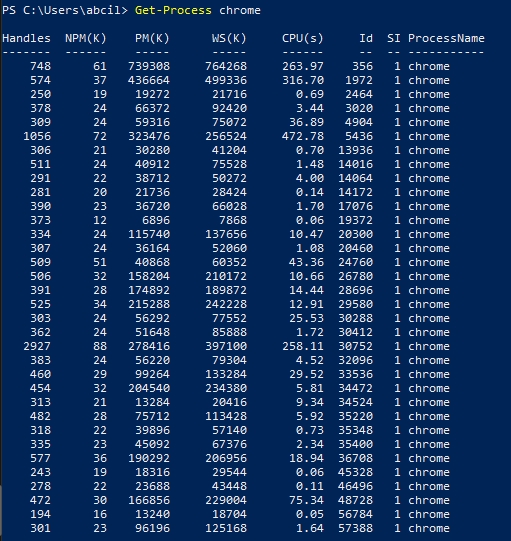
CPU(s): The amount of processor time that the process has used, in seconds.

Id: The process ID of the process.

SI: Session Id of the process.

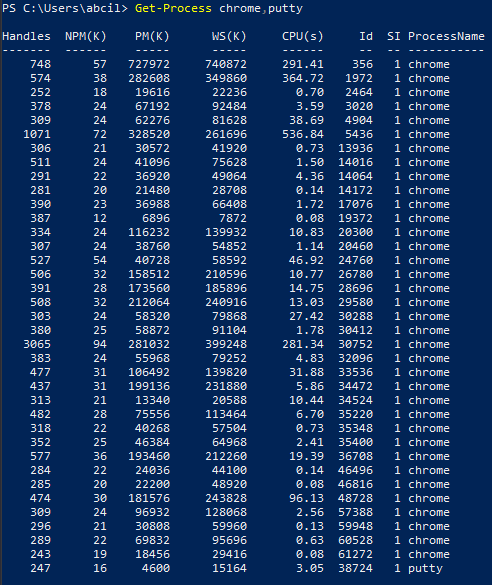
ProcessName: The name of the process.

1. 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 show the screenshot below.



1. Using this same command describe with example how you can get information about multiple processes

If you want to get information about both the chrome and putty processes at the same time, you would run the following command:  
Get-Process chrome,putty



This command will output a table with information for each of these processes.

1. 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





1. 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.

**Wait-Process**:

The Wait-Process blocks the PowerShell script from running until the specified process is stopped. The script will pause and wait until the putty process is stopped before continuing.

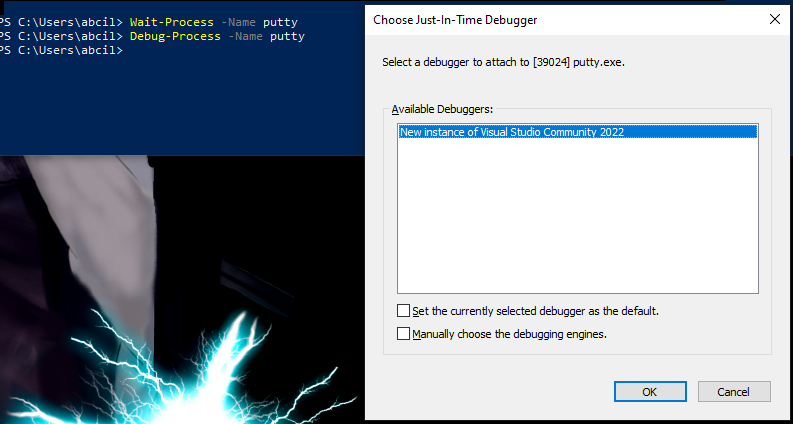




Putty was closed so the script unpaused.

**Debug-Process:**

The Debug-Process cmdlet is used to attach a debugger to a specified running process. In this example, a debugger would be attached to the putty process. This command requires administrative privileges and is typically used by developers and system administrators for troubleshooting purposes.



**Further Practice Questions.**

Answer the following questions based on your knowledge of process management in Linux.

1. What is a process in Linux? Answer: A process in Linux is a running instance of a program. Each process has its unique process ID (PID), and may have its own state, memory, and associated resources
2. Name three different states a process can be in, and briefly describe each state

a) Running

Description: The process is either running or ready to run

b) Sleeping

Description: The process is waiting for some condition to be true

c) Zombie

Description: The process has terminated, but information about the process is still in the process table

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

Example: ps -aux to list all running processes with detailed information

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

a) PID: Process ID, a unique identifier for each process

b) CPU%: The percentage of CPU time the process is consuming

c) MEM%: The percentage of memory the process is consuming

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

Method 1: Using the kill command followed by the PID, e.g., kill 69420

Method 2: Using the pkill command followed by the process name, e.g., pkill putty

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

Purpose of top: To provide a real-time updating view of the system's running processes, displaying a variety of process-related data

Sorting processes in top: Press the < or > keys to move the sort field left or right, respectively

1. Why is it important to exercise caution when terminating processes in Linux? Explain briefly.

Terminating processes recklessly can lead to data loss, system instability, or other undesired behaviors, especially if system or critical processes are terminated

1. Briefly explain the difference between the kill and killall commands in Linux.

Kill terminates processes by PID, while killall terminates processes by name, terminating all instances of the specified process

1. True or False: Terminating a process with SIGKILL allows it to perform cleanup operations before termination. Answer: False
2. Name two signals that can be sent to a process using the kill command, and briefly describe their effects.

Signal 1: SIGTERM (15)

Effect: Requests a process to terminate gracefully, allowing cleanup operations

Signal 2: SIGKILL (9)

Effect: Forces a process to terminate immediately, without any cleanup