OPS102 – Week 5 – Process Management - Sample Lab

Student Name: Chetan Arora

Student ID: 100976240

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:

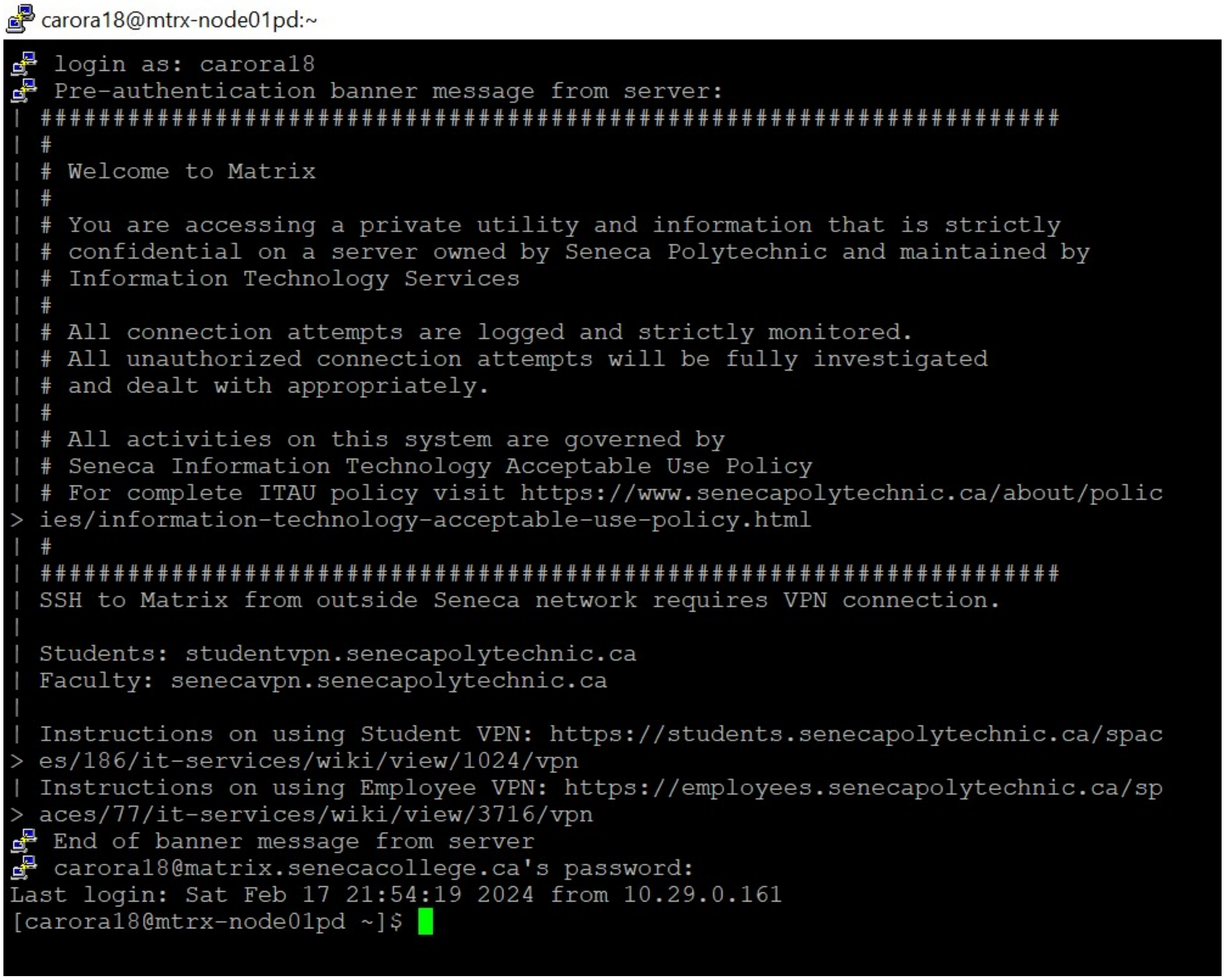
- **Processes and Process IDs (PIDs):** Every process in Linux or Windows is assigned a unique identifier called a Process ID (PID). PIDs enable the system to track and manage processes effectively. You can view the PIDs of running processes using various commands and utilities.
- **Process States**: 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

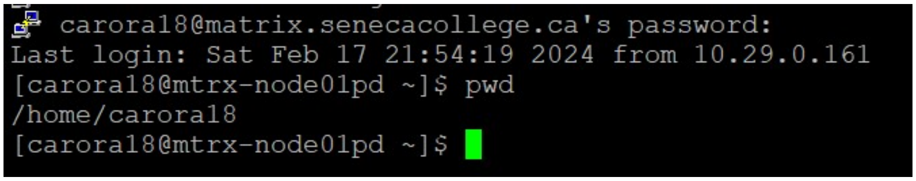
Perform the following steps:

1. Make certain that you are logged into your Matrix account



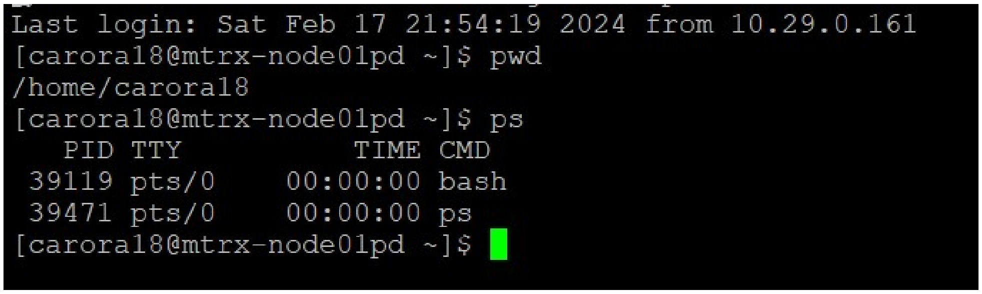
```
 login as: carora18
 Pre-authentication banner message from server:
| #######################################################################
| #
| # Welcome to Matrix
| #
| # You are accessing a private utility and information that is strictly
| # confidential on a server owned by Seneca Polytechnic and maintained by
| # Information Technology Services
| #
| # All connection attempts are logged and strictly monitored.
| # All unauthorized connection attempts will be fully investigated
| # and dealt with appropriately.
| #
| # All activities on this system are governed by
| # Seneca Information Technology Acceptable Use Policy
| # For complete ITAU policy visit https://www.senecapolytechnic.ca/about/polic
> ies/information-technology-acceptable-use-policy.html
| #
| #######################################################################
| SSH to Matrix from outside Seneca network requires VPN connection.
|
| Students: studentvpn.senecapolytechnic.ca
| Faculty: senecavpn.senecapolytechnic.ca
|
| Instructions on using Student VPN: https://students.senecapolytechnic.ca/spac
> es/186/it-services/wiki/view/1024/vpn
| Instructions on using Employee VPN: https://employees.senecapolytechnic.ca/sp
> aces/77/it-services/wiki/view/3716/vpn
 End of banner message from server
 carora18@matrix.senecacollege.ca's password:
Last login: Sat Feb 17 21:54:19 2024 from 10.29.0.161
[carora18@mtrx-node01pd ~]$
```

2. Issue a Linux command to confirm that you are located in your **home** directory.



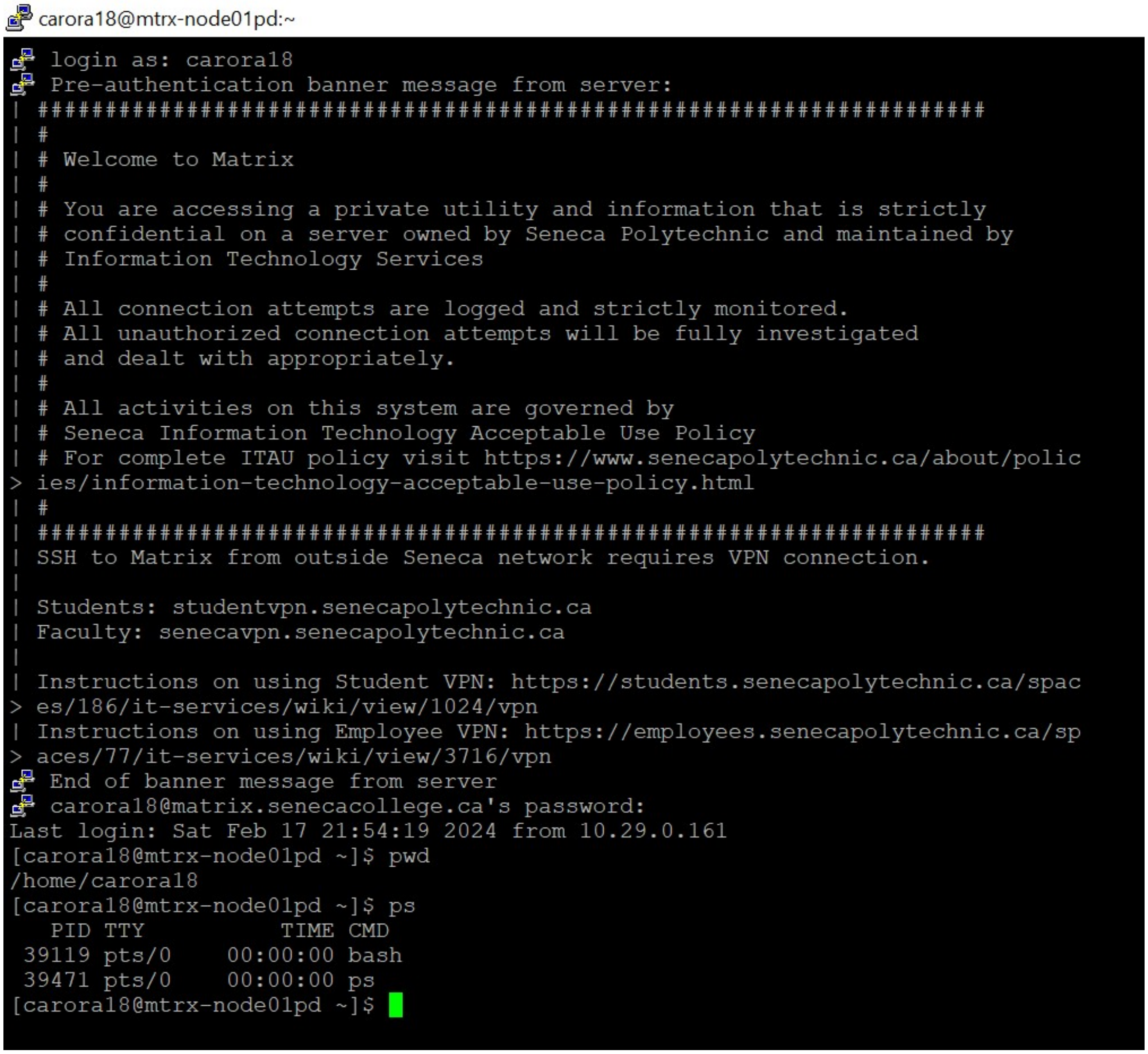
```
carora18@matrix.senecacollege.ca's password:
Last login: Sat Feb 17 21:54:19 2024 from 10.29.0.161
[carora18@mtrx-node01pd ~]$ pwd
/home/carora18
[carora18@mtrx-node01pd ~]$
```

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



```
Last login: Sat Feb 17 21:54:19 2024 from 10.29.0.161
[carora18@mtrx-node01pd ~]$ pwd
/home/carora18
[carora18@mtrx-node01pd ~]$ ps
   PID TTY          TIME CMD
 39119 pts/0    00:00:00 bash
 39471 pts/0    00:00:00 ps
[carora18@mtrx-node01pd ~]$
```

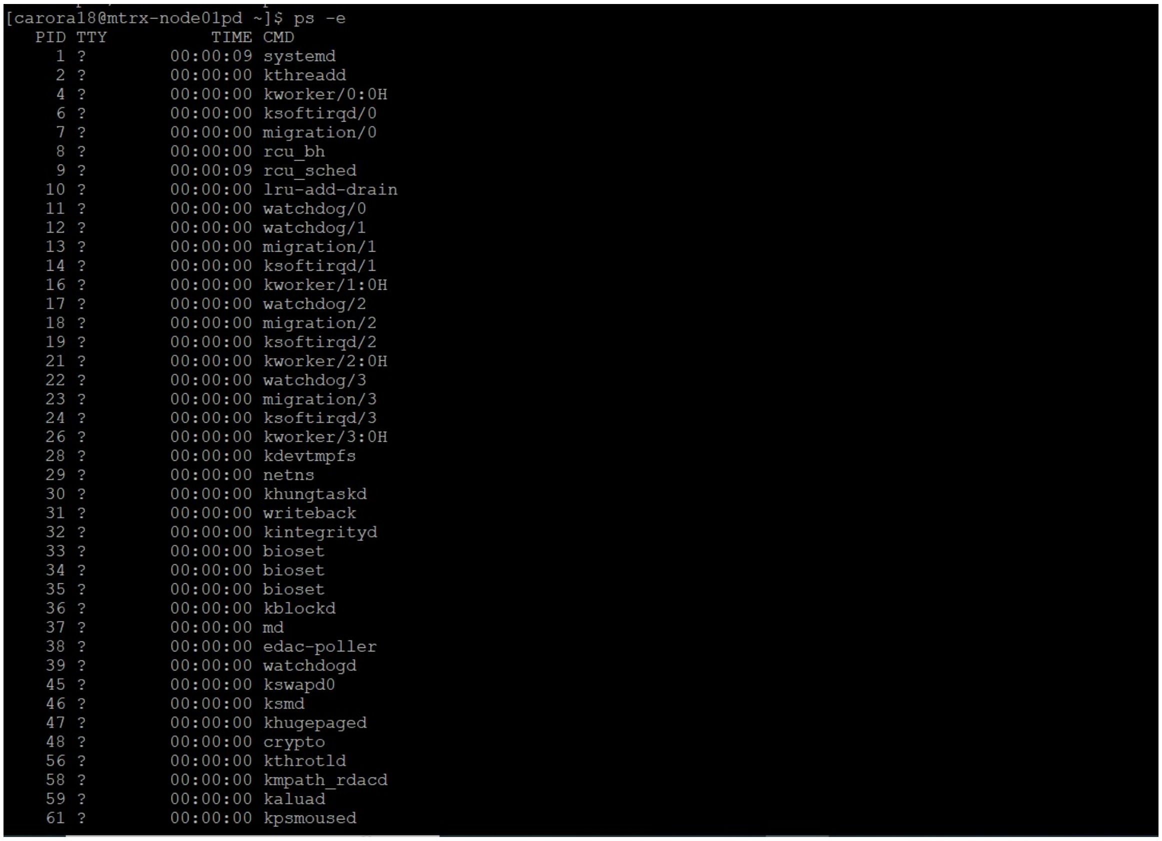
4. What output you see, take a screenshot and paste below.



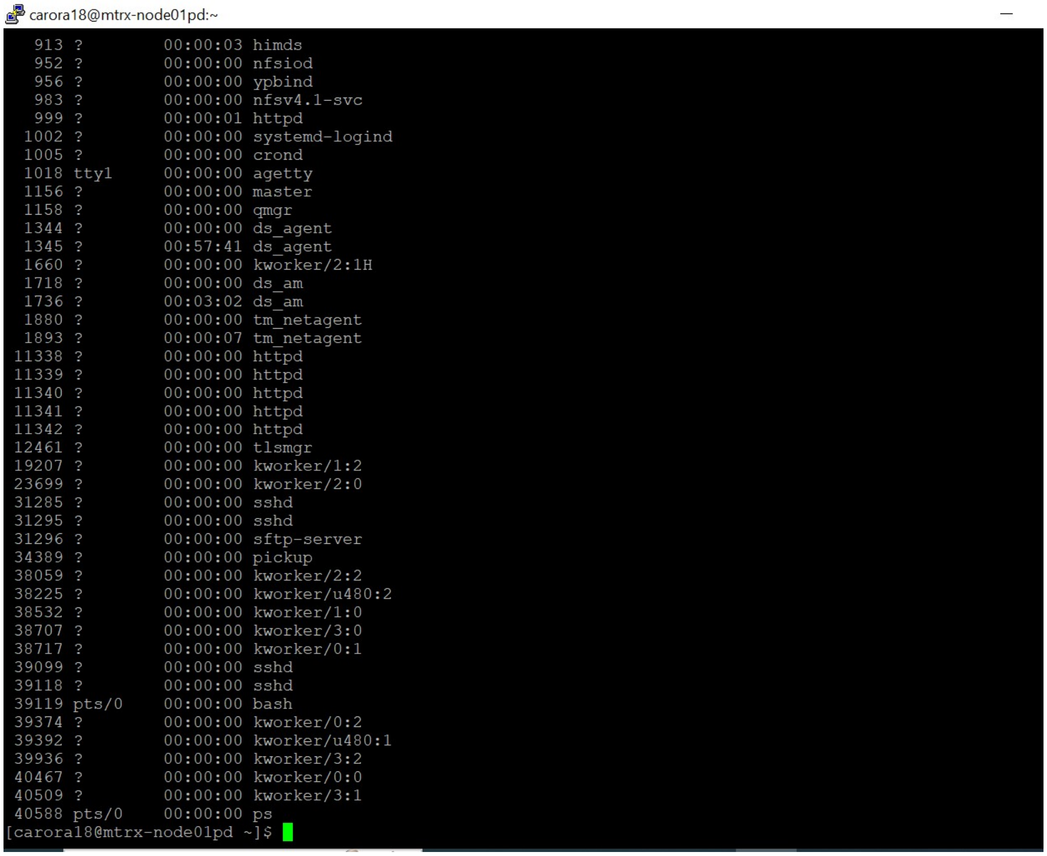
```
carora18@mtrx-node01pd:~
login as: carora18
Pre-authentication banner message from server:
| ######################################################################
| #
| # Welcome to Matrix
| #
| # You are accessing a private utility and information that is strictly
| # confidential on a server owned by Seneca Polytechnic and maintained by
| # Information Technology Services
| #
| # All connection attempts are logged and strictly monitored.
| # All unauthorized connection attempts will be fully investigated
| # and dealt with appropriately.
| #
| # All activities on this system are governed by
| # Seneca Information Technology Acceptable Use Policy
| # For complete ITAU policy visit https://www.senecapolytechnic.ca/about/polic
> ies/information-technology-acceptable-use-policy.html
| #
| ######################################################################
| SSH to Matrix from outside Seneca network requires VPN connection.
|
| Students: studentvpn.senecapolytechnic.ca
| Faculty: senecavpn.senecapolytechnic.ca
|
| Instructions on using Student VPN: https://students.senecapolytechnic.ca/spac
> es/186/it-services/wiki/view/1024/vpn
| Instructions on using Employee VPN: https://employees.senecapolytechnic.ca/sp
> aces/77/it-services/wiki/view/3716/vpn
End of banner message from server
carora18@matrix.senecacollege.ca's password:
Last login: Sat Feb 17 21:54:19 2024 from 10.29.0.161
[carora18@mtrx-node01pd ~]$ pwd
/home/carora18
[carora18@mtrx-node01pd ~]$ ps
   PID TTY          TIME CMD
 39119 pts/0    00:00:00 bash
 39471 pts/0    00:00:00 ps
[carora18@mtrx-node01pd ~]$
```

5. How many processes are currently running? What information is displayed for each process? Answer below.
   Ans: There are two processes currently running. The information displayed for each process are given below:-
   PID(Process ID):- A unique number that identifies the process.
   TTY(Terminal Type):- The name of the terminal where process is running.
   Time:- Cumulative execution time, the amount of CPU time used by the process.
   CMD:- The name of the executable file.

6. Use the ps command with the '-e' option to display information about all processes in the system. Run the command **ps -e**



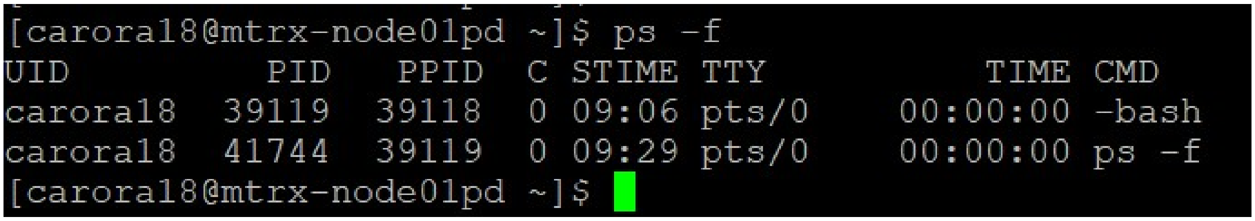
```
[carora18@mtrx-node01pd ~]$ ps -e
  PID TTY          TIME CMD
    1 ?        00:00:09 systemd
    2 ?        00:00:00 kthreadd
    4 ?        00:00:00 kworker/0:0H
    6 ?        00:00:00 ksoftirqd/0
    7 ?        00:00:00 migration/0
    8 ?        00:00:00 rcu_bh
    9 ?        00:00:09 rcu_sched
   10 ?        00:00:00 lru-add-drain
   11 ?        00:00:00 watchdog/0
   12 ?        00:00:00 watchdog/1
   13 ?        00:00:00 migration/1
   14 ?        00:00:00 ksoftirqd/1
   16 ?        00:00:00 kworker/1:0H
   17 ?        00:00:00 watchdog/2
   18 ?        00:00:00 migration/2
   19 ?        00:00:00 ksoftirqd/2
   21 ?        00:00:00 kworker/2:0H
   22 ?        00:00:00 watchdog/3
   23 ?        00:00:00 migration/3
   24 ?        00:00:00 ksoftirqd/3
   26 ?        00:00:00 kworker/3:0H
   28 ?        00:00:00 kdevtmpfs
   29 ?        00:00:00 netns
   30 ?        00:00:00 khungtaskd
   31 ?        00:00:00 writeback
   32 ?        00:00:00 kintegrityd
   33 ?        00:00:00 bioset
   34 ?        00:00:00 bioset
   35 ?        00:00:00 bioset
   36 ?        00:00:00 kblockd
   37 ?        00:00:00 md
   38 ?        00:00:00 edac-poller
   39 ?        00:00:00 watchdogd
   45 ?        00:00:00 kswapd0
   46 ?        00:00:00 ksmd
   47 ?        00:00:00 khugepaged
   48 ?        00:00:00 crypto
   56 ?        00:00:00 kthrotld
   58 ?        00:00:00 kmpath_rdacd
   59 ?        00:00:00 kaluad
   61 ?        00:00:00 kpsmoused
```



```
   913 ?        00:00:03 himds
   952 ?        00:00:00 nfsiod
   956 ?        00:00:00 ypbind
   983 ?        00:00:00 nfsv4.1-svc
   999 ?        00:00:01 httpd
  1002 ?        00:00:00 systemd-logind
  1005 ?        00:00:00 crond
  1018 tty1     00:00:00 agetty
  1156 ?        00:00:00 master
  1158 ?        00:00:00 qmgr
  1344 ?        00:00:00 ds_agent
  1345 ?        00:57:41 ds_agent
  1660 ?        00:00:00 kworker/2:1H
  1718 ?        00:00:00 ds_am
  1736 ?        00:03:02 ds_am
  1880 ?        00:00:00 tm_netagent
  1893 ?        00:00:07 tm_netagent
 11338 ?        00:00:00 httpd
 11339 ?        00:00:00 httpd
 11340 ?        00:00:00 httpd
 11341 ?        00:00:00 httpd
 11342 ?        00:00:00 httpd
 12461 ?        00:00:00 tlsmgr
 19207 ?        00:00:00 kworker/1:2
 23699 ?        00:00:00 kworker/2:0
 31285 ?        00:00:00 sshd
 31295 ?        00:00:00 sshd
 31296 ?        00:00:00 sftp-server
 34389 ?        00:00:00 pickup
 38059 ?        00:00:00 kworker/2:2
 38225 ?        00:00:00 kworker/u480:2
 38532 ?        00:00:00 kworker/1:0
 38707 ?        00:00:00 kworker/3:0
 38717 ?        00:00:00 kworker/0:1
 39099 ?        00:00:00 sshd
 39118 ?        00:00:00 sshd
 39119 pts/0    00:00:00 bash
 39374 ?        00:00:00 kworker/0:2
 39392 ?        00:00:00 kworker/u480:1
 39936 ?        00:00:00 kworker/3:2
 40467 ?        00:00:00 kworker/0:0
 40509 ?        00:00:00 kworker/3:1
 40588 pts/0    00:00:00 ps
[carora18@mtrx-node01pd ~]$
```

7. Analyze the output and identify the running processes on your system. Note the PID, TTY, and CMD columns. What do these column mean?
   Ans:- PID(Process ID):- A unique number that identifies the process.
   TTY(Terminal Type):- The name of the terminal where process is running.
   Time:- Cumulative execution time, the amount of CPU time used by the process.
   CMD:- The name of the executable file or command.
   For example: PID 1 is associated with the systemd process, which is the parent process for all other processes.
   By analyzing the output, we can see the currently running processes on the system and their characteristics.

8. Use the 'ps' command with the '-f' option to display a full-format listing of the processes. Run the command **`ps -f`**

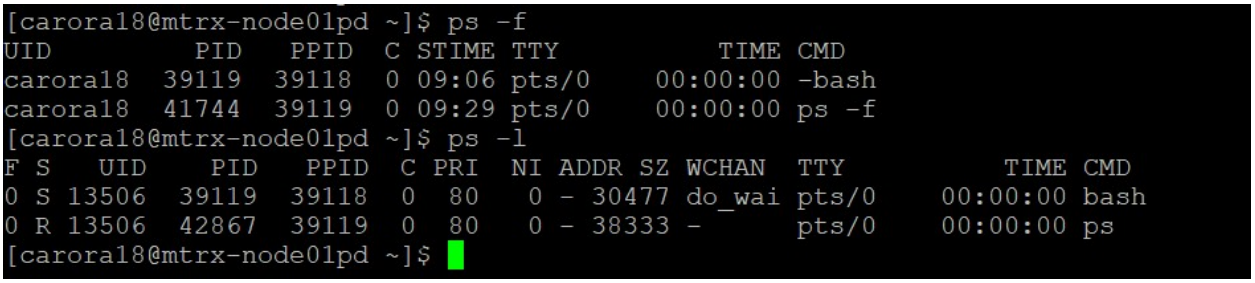


```
[carora18@mtrx-node01pd ~]$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
carora18   39119   39118  0 09:06 pts/0    00:00:00 -bash
carora18   41744   39119  0 09:29 pts/0    00:00:00 ps -f
[carora18@mtrx-node01pd ~]$
```

9. Examine the output, which provides detailed information about each process, including UID, PID, PPID, CPU%, MEM%, START, and CMD
   Ans:- By analyzing the output, we can see the currently running processes on the system by the user. Here is what each column means:

   UID(User ID):- the name or number of the user who owns the process.
   PID(Process ID):- A unique number that identifies the process.
   PPID(Parent Process ID):- the PID of the process that started this process.
   C:- CPU utilization, the percentage of CPU time used by the process in the last second.
   STIME:- Start Time, the time when the process was started.
   TTY(Terminal Type):- The name of the terminal where process is running.
   Time:- Cumulative execution time, the amount of CPU time used by the process.
   CMD:- The name of the executable file or command.

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



```
[carora18@mtrx-node01pd ~]$ ps -f
UID          PID    PPID  C STIME TTY          TIME CMD
carora18   39119   39118  0 09:06 pts/0    00:00:00 -bash
carora18   41744   39119  0 09:29 pts/0    00:00:00 ps -f
[carora18@mtrx-node01pd ~]$ ps -l
F S   UID    PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S 13506  39119   39118  0  80   0 - 30477 do_wai pts/0    00:00:00 bash
0 R 13506  42867   39119  0  80   0 - 38333 -      pts/0    00:00:00 ps
[carora18@mtrx-node01pd ~]$
```

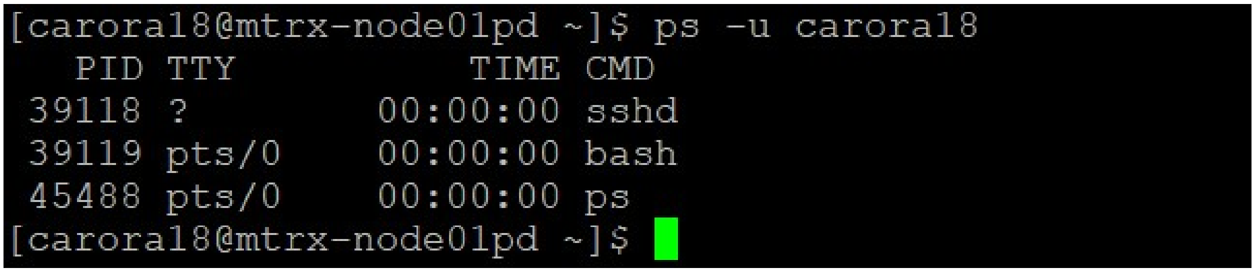
11. Analyze the output and observe the columns displayed, including F, S, UID, PID, PPID, PRI, NI, ADDR, SZ, RSS, WCHAN, STAT, TTY, TIME, and CMD.

   Ans:- By analyzing the output, we can see the currently running processes on the system by the user. Here is what each column means:

   F(Flags):- a set of hexadecimal digits that indicate the status of the process, such as whether it is running in the foreground or background, whether it is stopped or traced, etc.
   S(State):- a single letter that indicates the current state of the process, such as R (running), S (sleeping), T (stopped), Z (zombie), etc.
   UID(User ID):- the name or number of the user who owns the process.
   PID(Process ID):- A unique number that identifies the process.
   PPID(Parent Process ID):- the PID of the process that started this process.
   C:- CPU utilization, the percentage of CPU time used by the process in the last second
   PRI(Priority):- a number that indicates the scheduling priority of the process, with lower numbers having higher priority

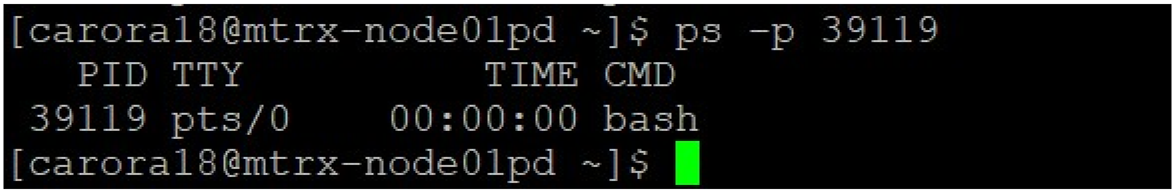
NI(Nice):- a number that influences the scheduling priority of the process, with lower numbers having higher priority
ADDR(Address):- the memory address of the process
SZ(Size):- the size in physical pages of the core image of the process
RSS(Resident Set Size):- the amount of physical memory used by the process
WCHAN(Waiting Channel):- the name of the kernel function or event on which the process is waiting, or '-' if it is running
TTY(Terminal Type):- The name of the terminal where process is running.
Time:- Cumulative execution time, the amount of CPU time used by the process.
CMD:- The name of the executable file or command.

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



```
[carora18@mtrx-node01pd ~]$ ps -u carora18
   PID TTY          TIME CMD
 39118 ?        00:00:00 sshd
 39119 pts/0    00:00:00 bash
 45488 pts/0    00:00:00 ps
[carora18@mtrx-node01pd ~]$ 
```

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



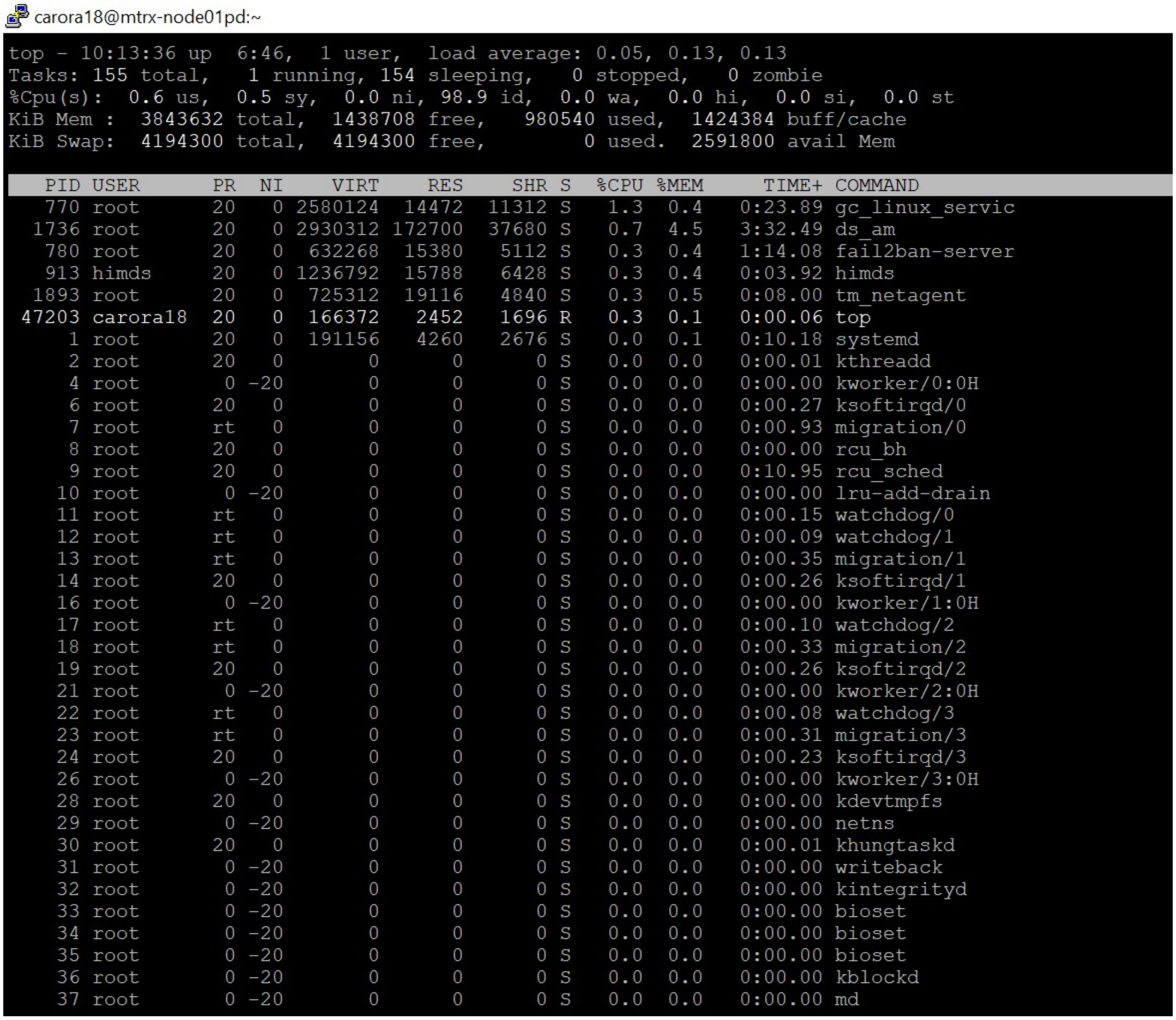
```
[carora18@mtrx-node01pd ~]$ ps -p 39119
   PID TTY          TIME CMD
 39119 pts/0    00:00:00 bash
[carora18@mtrx-node01pd ~]$ 
```

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?



carora18@mtrx-node01pd:~

```
top - 10:13:36 up  6:46,  1 user,  load average: 0.05, 0.13, 0.13
Tasks: 155 total,   1 running, 154 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.6 us,  0.5 sy,  0.0 ni, 98.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  3843632 total,  1438708 free,   980540 used,  1424384 buff/cache
KiB Swap:  4194300 total,  4194300 free,        0 used.  2591800 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
   770 root      20   0 2580124  14472  11312 S   1.3  0.4   0:23.89 gc_linux_servic
  1736 root      20   0 2930312 172700  37680 S   0.7  4.5   3:32.49 ds_am
   780 root      20   0  632268  15380   5112 S   0.3  0.4   1:14.08 fail2ban-server
   913 himds     20   0 1236792  15788   6428 S   0.3  0.4   0:03.92 himds
  1893 root      20   0  725312  19116   4840 S   0.3  0.5   0:08.00 tm_netagent
 47203 carora18  20   0  166372   2452   1696 R   0.3  0.1   0:00.06 top
     1 root      20   0  191156   4260   2676 S   0.0  0.1   0:10.18 systemd
     2 root      20   0       0      0      0 S   0.0  0.0   0:00.01 kthreadd
     4 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
     6 root      20   0       0      0      0 S   0.0  0.0   0:00.27 ksoftirqd/0
     7 root      rt   0       0      0      0 S   0.0  0.0   0:00.93 migration/0
     8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
     9 root      20   0       0      0      0 S   0.0  0.0   0:10.95 rcu_sched
    10 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 lru-add-drain
    11 root      rt   0       0      0      0 S   0.0  0.0   0:00.15 watchdog/0
    12 root      rt   0       0      0      0 S   0.0  0.0   0:00.09 watchdog/1
    13 root      rt   0       0      0      0 S   0.0  0.0   0:00.35 migration/1
    14 root      20   0       0      0      0 S   0.0  0.0   0:00.26 ksoftirqd/1
    16 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:0H
    17 root      rt   0       0      0      0 S   0.0  0.0   0:00.10 watchdog/2
    18 root      rt   0       0      0      0 S   0.0  0.0   0:00.33 migration/2
    19 root      20   0       0      0      0 S   0.0  0.0   0:00.26 ksoftirqd/2
    21 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/2:0H
    22 root      rt   0       0      0      0 S   0.0  0.0   0:00.08 watchdog/3
    23 root      rt   0       0      0      0 S   0.0  0.0   0:00.31 migration/3
    24 root      20   0       0      0      0 S   0.0  0.0   0:00.23 ksoftirqd/3
    26 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/3:0H
    28 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kdevtmpfs
    29 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 netns
    30 root      20   0       0      0      0 S   0.0  0.0   0:00.01 khungtaskd
    31 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 writeback
    32 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kintegrityd
    33 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
    34 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
    35 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
    36 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kblockd
    37 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 md
```

2. Once the top command is running, you'll see a continuously updated display with various sections and columns.

   Ans:- Yes , I can see 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+

Ans:- By analyzing the output. Here is what each column means:

a. PR: This column shows the priority of the task, which is a number that indicates how urgently the task needs the CPU. The lower the number, the higher the priority.

b. NI: This column displays the nice value of a task, which is a user-space concept that allows you to prioritize tasks. The nice value ranges from -20 (highest priority) to 19 (lowest priority). A positive nice value means that the task is being nice and letting other tasks use the CPU more.

c. VIRT: This column indicates how much virtual memory is associated with the process, which is the total amount of memory that the process can access, including physical memory, swap space, and memory mapped files.

d. RES: This column shows the resident size of data, which is the portion of a process's memory that is held in RAM. This is the actual memory usage of the process, excluding the memory that is swapped out or shared with other processes.

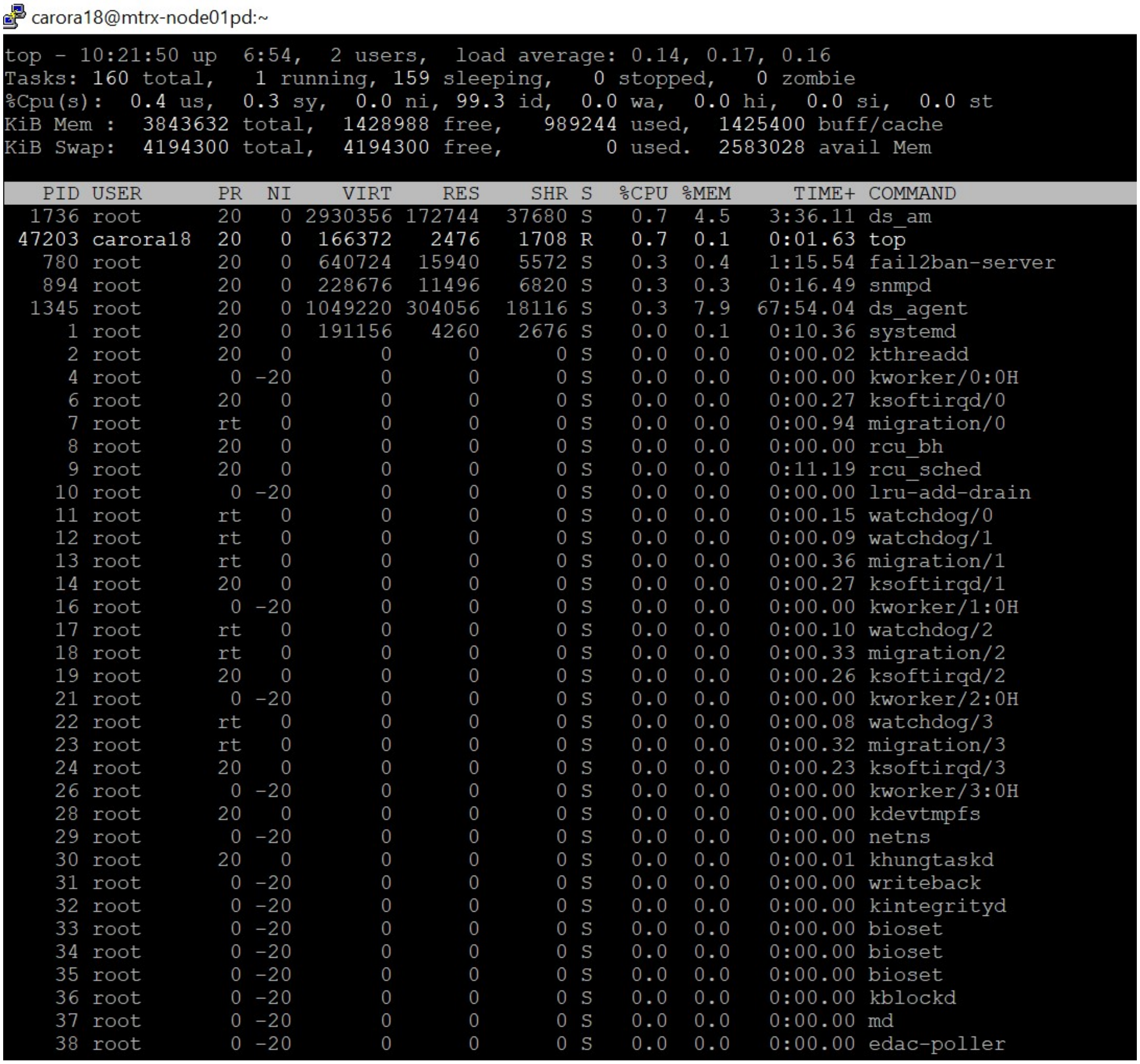
e. %CPU: This column represents CPU usage, indicating how much CPU time (in percentage) the process is consuming. This is calculated by dividing the CPU time used by the process by the total CPU time available in the system.

f. %MEM: This column indicates how much memory (in percentage) from RAM is being used by the process. This is calculated by dividing the resident size of the process by the total physical memory available in the system.

g. TIME+: This column shows the total execution time for the task since it started, in the format of minutes:seconds.hundredths. This is the cumulative amount of CPU time that the process has used.

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.

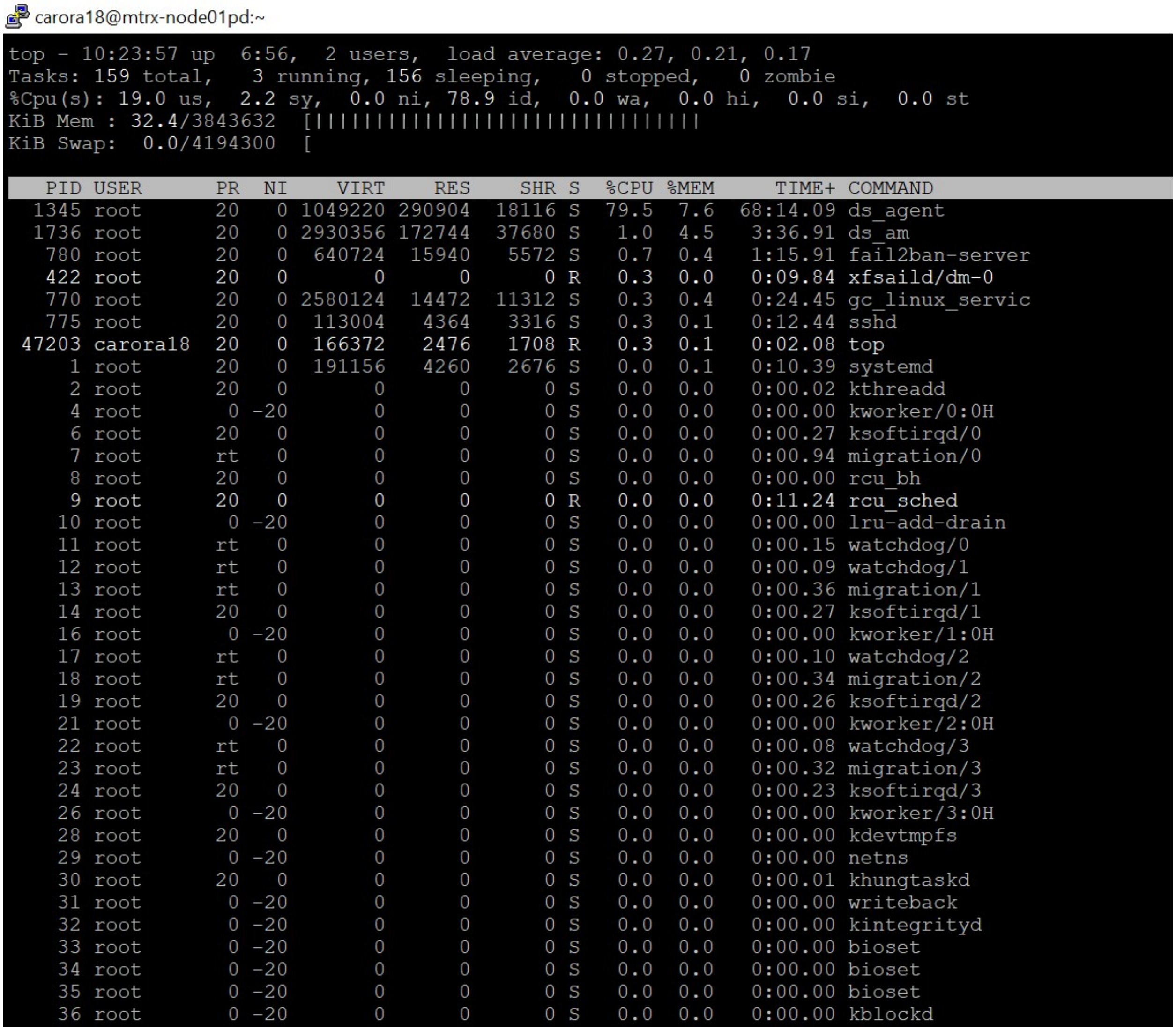
   After Pressing ‘P’



```
top - 10:21:50 up  6:54,  2 users,  load average: 0.14, 0.17, 0.16
Tasks: 160 total,   1 running, 159 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.4 us,  0.3 sy,  0.0 ni, 99.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  3843632 total,  1428988 free,   989244 used,  1425400 buff/cache
KiB Swap:  4194300 total,  4194300 free,        0 used.  2583028 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1736 root      20   0 2930356 172744  37680 S   0.7  4.5   3:36.11 ds_am
47203 carora18  20   0  166372   2476   1708 R   0.7  0.1   0:01.63 top
  780 root      20   0  640724  15940   5572 S   0.3  0.4   1:15.54 fail2ban-server
  894 root      20   0  228676  11496   6820 S   0.3  0.3   0:16.49 snmpd
 1345 root      20   0 1049220 304056  18116 S   0.3  7.9  67:54.04 ds_agent
    1 root      20   0  191156   4260   2676 S   0.0  0.1   0:10.36 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.02 kthreadd
    4 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S   0.0  0.0   0:00.27 ksoftirqd/0
    7 root      rt   0       0      0      0 S   0.0  0.0   0:00.94 migration/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0   0:11.19 rcu_sched
   10 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 lru-add-drain
   11 root      rt   0       0      0      0 S   0.0  0.0   0:00.15 watchdog/0
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.09 watchdog/1
   13 root      rt   0       0      0      0 S   0.0  0.0   0:00.36 migration/1
   14 root      20   0       0      0      0 S   0.0  0.0   0:00.27 ksoftirqd/1
   16 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:0H
   17 root      rt   0       0      0      0 S   0.0  0.0   0:00.10 watchdog/2
   18 root      rt   0       0      0      0 S   0.0  0.0   0:00.33 migration/2
   19 root      20   0       0      0      0 S   0.0  0.0   0:00.26 ksoftirqd/2
   21 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/2:0H
   22 root      rt   0       0      0      0 S   0.0  0.0   0:00.08 watchdog/3
   23 root      rt   0       0      0      0 S   0.0  0.0   0:00.32 migration/3
   24 root      20   0       0      0      0 S   0.0  0.0   0:00.23 ksoftirqd/3
   26 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/3:0H
   28 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kdevtmpfs
   29 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 netns
   30 root      20   0       0      0      0 S   0.0  0.0   0:00.01 khungtaskd
   31 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 writeback
   32 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kintegrityd
   33 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   34 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   35 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   36 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kblockd
   37 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 md
   38 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 edac-poller
```

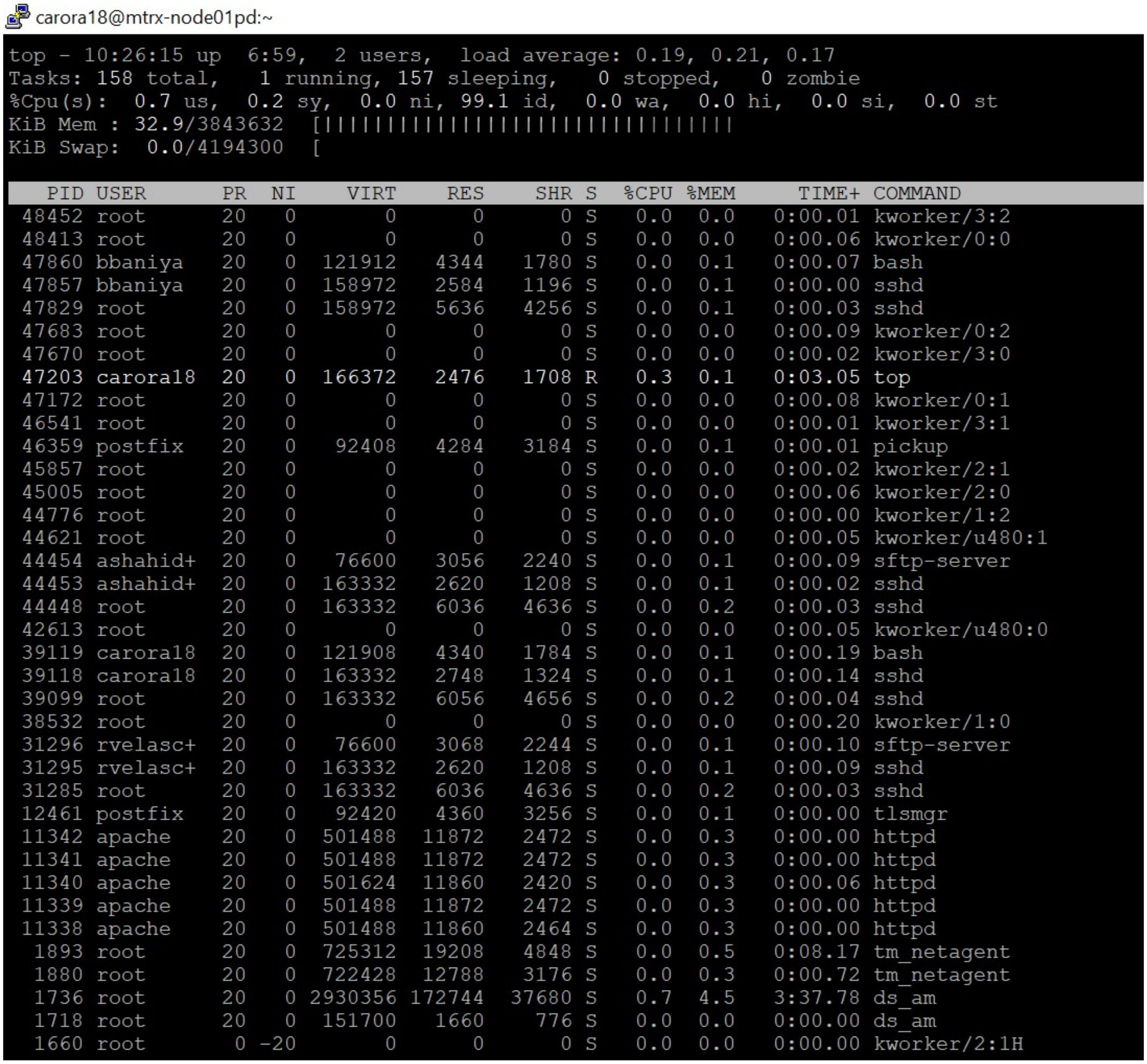
After Pressing ‘M’



```
top - 10:23:57 up  6:56,  2 users,  load average: 0.27, 0.21, 0.17
Tasks: 159 total,   3 running, 156 sleeping,   0 stopped,   0 zombie
%Cpu(s): 19.0 us,  2.2 sy,  0.0 ni, 78.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32.4/3843632  [||||||||||||||||||||||||||||||||
KiB Swap:  0.0/4194300  [

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1345 root      20   0 1049220 290904  18116 S  79.5  7.6  68:14.09 ds_agent
 1736 root      20   0 2930356 172744  37680 S   1.0  4.5   3:36.91 ds_am
  780 root      20   0  640724  15940   5572 S   0.7  0.4   1:15.91 fail2ban-server
  422 root      20   0       0      0      0 R   0.3  0.0   0:09.84 xfsaild/dm-0
  770 root      20   0 2580124  14472  11312 S   0.3  0.4   0:24.45 gc_linux_servic
  775 root      20   0  113004   4364   3316 S   0.3  0.1   0:12.44 sshd
47203 carora18  20   0  166372   2476   1708 R   0.3  0.1   0:02.08 top
    1 root      20   0  191156   4260   2676 S   0.0  0.1   0:10.39 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.02 kthreadd
    4 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    6 root      20   0       0      0      0 S   0.0  0.0   0:00.27 ksoftirqd/0
    7 root      rt   0       0      0      0 S   0.0  0.0   0:00.94 migration/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 R   0.0  0.0   0:11.24 rcu_sched
   10 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 lru-add-drain
   11 root      rt   0       0      0      0 S   0.0  0.0   0:00.15 watchdog/0
   12 root      rt   0       0      0      0 S   0.0  0.0   0:00.09 watchdog/1
   13 root      rt   0       0      0      0 S   0.0  0.0   0:00.36 migration/1
   14 root      20   0       0      0      0 S   0.0  0.0   0:00.27 ksoftirqd/1
   16 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1:0H
   17 root      rt   0       0      0      0 S   0.0  0.0   0:00.10 watchdog/2
   18 root      rt   0       0      0      0 S   0.0  0.0   0:00.34 migration/2
   19 root      20   0       0      0      0 S   0.0  0.0   0:00.26 ksoftirqd/2
   21 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/2:0H
   22 root      rt   0       0      0      0 S   0.0  0.0   0:00.08 watchdog/3
   23 root      rt   0       0      0      0 S   0.0  0.0   0:00.32 migration/3
   24 root      20   0       0      0      0 S   0.0  0.0   0:00.23 ksoftirqd/3
   26 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/3:0H
   28 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kdevtmpfs
   29 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 netns
   30 root      20   0       0      0      0 S   0.0  0.0   0:00.01 khungtaskd
   31 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 writeback
   32 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kintegrityd
   33 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   34 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   35 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
   36 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kblockd
```

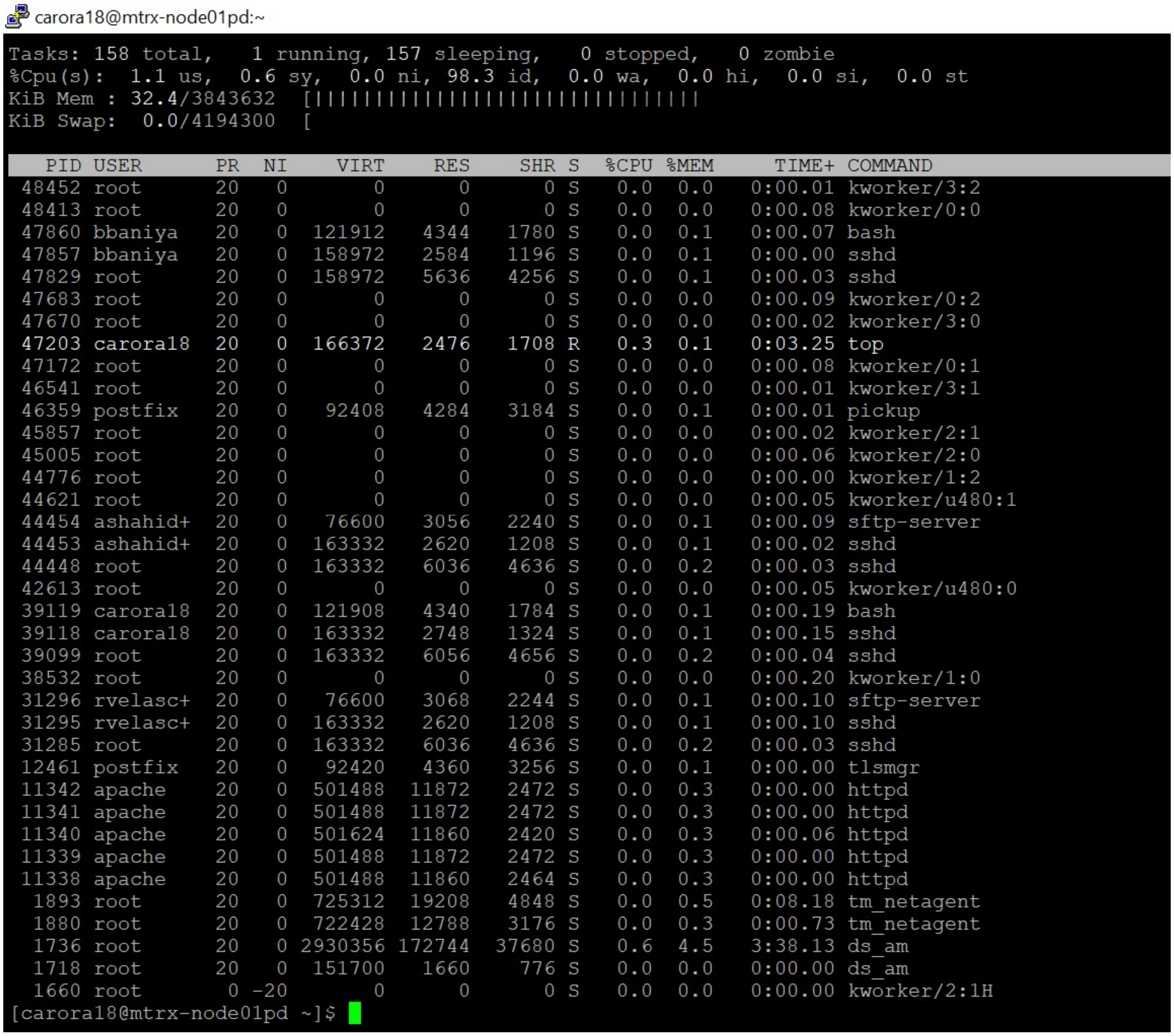
After Pressing ‘N’



```
top - 10:26:15 up  6:59,  2 users,  load average: 0.19, 0.21, 0.17
Tasks: 158 total,   1 running, 157 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.7 us,  0.2 sy,  0.0 ni, 99.1 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32.9/3843632  [|||||||||||||||||||||||||||||||||
KiB Swap:  0.0/4194300  [

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 48452 root      20   0       0      0      0 S   0.0  0.0   0:00.01 kworker/3:2
 48413 root      20   0       0      0      0 S   0.0  0.0   0:00.06 kworker/0:0
 47860 bbaniya   20   0  121912   4344   1780 S   0.0  0.1   0:00.07 bash
 47857 bbaniya   20   0  158972   2584   1196 S   0.0  0.1   0:00.00 sshd
 47829 root      20   0  158972   5636   4256 S   0.0  0.1   0:00.03 sshd
 47683 root      20   0       0      0      0 S   0.0  0.0   0:00.09 kworker/0:2
 47670 root      20   0       0      0      0 S   0.0  0.0   0:00.02 kworker/3:0
 47203 carora18  20   0  166372   2476   1708 R   0.3  0.1   0:03.05 top
 47172 root      20   0       0      0      0 S   0.0  0.0   0:00.08 kworker/0:1
 46541 root      20   0       0      0      0 S   0.0  0.0   0:00.01 kworker/3:1
 46359 postfix   20   0   92408   4284   3184 S   0.0  0.1   0:00.01 pickup
 45857 root      20   0       0      0      0 S   0.0  0.0   0:00.02 kworker/2:1
 45005 root      20   0       0      0      0 S   0.0  0.0   0:00.06 kworker/2:0
 44776 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kworker/1:2
 44621 root      20   0       0      0      0 S   0.0  0.0   0:00.05 kworker/u480:1
 44454 ashahid+  20   0   76600   3056   2240 S   0.0  0.1   0:00.09 sftp-server
 44453 ashahid+  20   0  163332   2620   1208 S   0.0  0.1   0:00.02 sshd
 44448 root      20   0  163332   6036   4636 S   0.0  0.2   0:00.03 sshd
 42613 root      20   0       0      0      0 S   0.0  0.0   0:00.05 kworker/u480:0
 39119 carora18  20   0  121908   4340   1784 S   0.0  0.1   0:00.19 bash
 39118 carora18  20   0  163332   2748   1324 S   0.0  0.1   0:00.14 sshd
 39099 root      20   0  163332   6056   4656 S   0.0  0.2   0:00.04 sshd
 38532 root      20   0       0      0      0 S   0.0  0.0   0:00.20 kworker/1:0
 31296 rvelasc+  20   0   76600   3068   2244 S   0.0  0.1   0:00.10 sftp-server
 31295 rvelasc+  20   0  163332   2620   1208 S   0.0  0.1   0:00.09 sshd
 31285 root      20   0  163332   6036   4636 S   0.0  0.2   0:00.03 sshd
 12461 postfix   20   0   92420   4360   3256 S   0.0  0.1   0:00.00 tlsmgr
 11342 apache    20   0  501488  11872   2472 S   0.0  0.3   0:00.00 httpd
 11341 apache    20   0  501488  11872   2472 S   0.0  0.3   0:00.00 httpd
 11340 apache    20   0  501624  11860   2420 S   0.0  0.3   0:00.06 httpd
 11339 apache    20   0  501488  11872   2472 S   0.0  0.3   0:00.00 httpd
 11338 apache    20   0  501488  11860   2464 S   0.0  0.3   0:00.00 httpd
  1893 root      20   0  725312  19208   4848 S   0.0  0.5   0:08.17 tm_netagent
  1880 root      20   0  722428  12788   3176 S   0.0  0.3   0:00.72 tm_netagent
  1736 root      20   0 2930356 172744  37680 S   0.7  4.5   3:37.78 ds_am
  1718 root      20   0  151700   1660    776 S   0.0  0.0   0:00.00 ds_am
  1660 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/2:1H
```

5. To exit the top command, simply press 'q'. This will close the top display and return you to the terminal prompt.



```
Tasks: 158 total,   1 running, 157 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.1 us,  0.6 sy,  0.0 ni, 98.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 32.4/3843632  [||||||||||||||||||||||||||||||||
KiB Swap:  0.0/4194300  [

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 48452 root      20   0       0      0      0 S   0.0  0.0   0:00.01 kworker/3:2
 48413 root      20   0       0      0      0 S   0.0  0.0   0:00.08 kworker/0:0
 47860 bbaniya   20   0  121912   4344   1780 S   0.0  0.1   0:00.07 bash
 47857 bbaniya   20   0  158972   2584   1196 S   0.0  0.1   0:00.00 sshd
 47829 root      20   0  158972   5636   4256 S   0.0  0.1   0:00.03 sshd
 47683 root      20   0       0      0      0 S   0.0  0.0   0:00.09 kworker/0:2
 47670 root      20   0       0      0      0 S   0.0  0.0   0:00.02 kworker/3:0
 47203 carora18  20   0  166372   2476   1708 R   0.3  0.1   0:03.25 top
 47172 root      20   0       0      0      0 S   0.0  0.0   0:00.08 kworker/0:1
 46541 root      20   0       0      0      0 S   0.0  0.0   0:00.01 kworker/3:1
 46359 postfix   20   0   92408   4284   3184 S   0.0  0.1   0:00.01 pickup
 45857 root      20   0       0      0      0 S   0.0  0.0   0:00.02 kworker/2:1
 45005 root      20   0       0      0      0 S   0.0  0.0   0:00.06 kworker/2:0
 44776 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kworker/1:2
 44621 root      20   0       0      0      0 S   0.0  0.0   0:00.05 kworker/u480:1
 44454 ashahid+  20   0   76600   3056   2240 S   0.0  0.1   0:00.09 sftp-server
 44453 ashahid+  20   0  163332   2620   1208 S   0.0  0.1   0:00.02 sshd
 44448 root      20   0  163332   6036   4636 S   0.0  0.2   0:00.03 sshd
 42613 root      20   0       0      0      0 S   0.0  0.0   0:00.05 kworker/u480:0
 39119 carora18  20   0  121908   4340   1784 S   0.0  0.1   0:00.19 bash
 39118 carora18  20   0  163332   2748   1324 S   0.0  0.1   0:00.15 sshd
 39099 root      20   0  163332   6056   4656 S   0.0  0.2   0:00.04 sshd
 38532 root      20   0       0      0      0 S   0.0  0.0   0:00.20 kworker/1:0
 31296 rvelasc+  20   0   76600   3068   2244 S   0.0  0.1   0:00.10 sftp-server
 31295 rvelasc+  20   0  163332   2620   1208 S   0.0  0.1   0:00.10 sshd
 31285 root      20   0  163332   6036   4636 S   0.0  0.2   0:00.03 sshd
 12461 postfix   20   0   92420   4360   3256 S   0.0  0.1   0:00.00 tlsmgr
 11342 apache    20   0  501488  11872   2472 S   0.0  0.3   0:00.00 httpd
 11341 apache    20   0  501488  11872   2472 S   0.0  0.3   0:00.00 httpd
 11340 apache    20   0  501624  11860   2420 S   0.0  0.3   0:00.06 httpd
 11339 apache    20   0  501488  11872   2472 S   0.0  0.3   0:00.00 httpd
 11338 apache    20   0  501488  11860   2464 S   0.0  0.3   0:00.00 httpd
  1893 root      20   0  725312  19208   4848 S   0.0  0.5   0:08.18 tm_netagent
  1880 root      20   0  722428  12788   3176 S   0.0  0.3   0:00.73 tm_netagent
  1736 root      20   0 2930356 172744  37680 S   0.6  4.5   3:38.13 ds_am
  1718 root      20   0  151700   1660    776 S   0.0  0.0   0:00.00 ds_am
  1660 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/2:1H
[carora18@mtrx-node01pd ~]$
```

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.

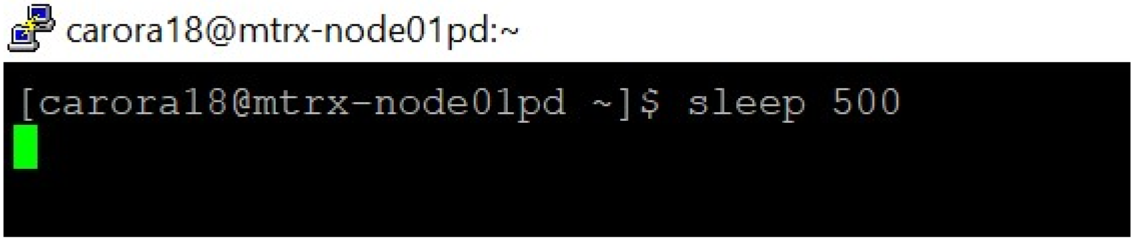
Common Signals: Linux systems have a set of standard signals defined, each with its own signal number. Some commonly used signals include:

- 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.

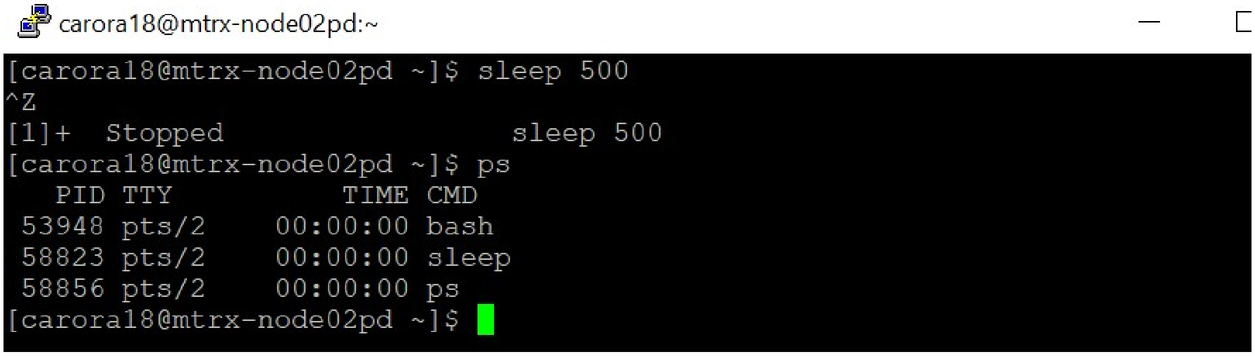
Perform the following steps:

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**.



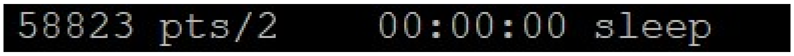
```
[carora18@mtrx-node01pd ~]$ sleep 500
█
```

2. Run the command: **ps**



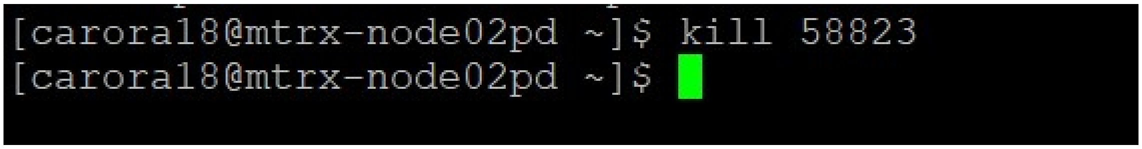
```
[carora18@mtrx-node02pd ~]$ sleep 500
^Z
[1]+  Stopped                 sleep 500
[carora18@mtrx-node02pd ~]$ ps
   PID TTY          TIME CMD
 53948 pts/2    00:00:00 bash
 58823 pts/2    00:00:00 sleep
 58856 pts/2    00:00:00 ps
[carora18@mtrx-node02pd ~]$ 
```

3. Note the process id of sleep command.
   ANS:-58823



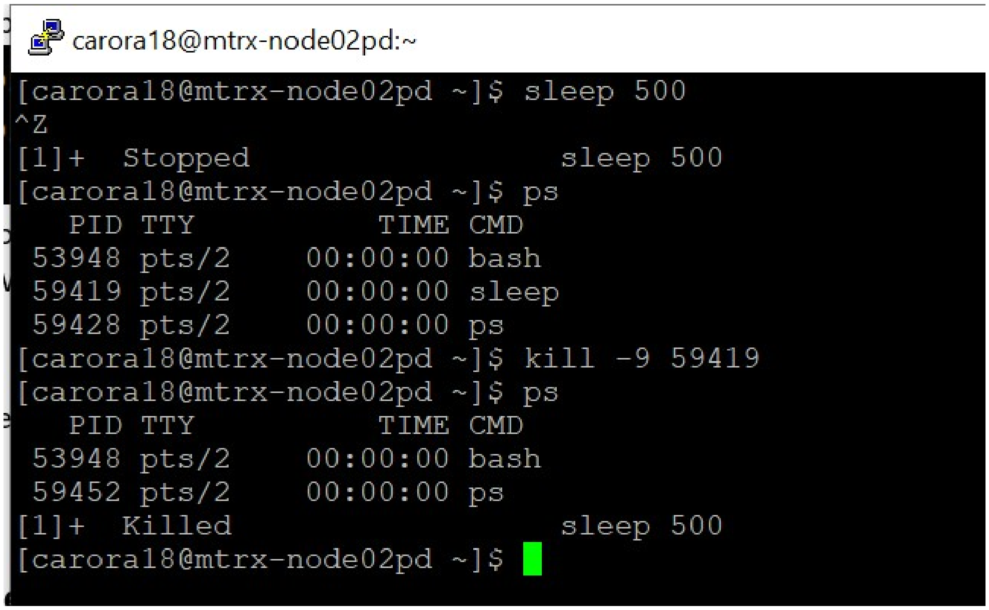
```
 58823 pts/2    00:00:00 sleep
```

4. 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.



```
[carora18@mtrx-node02pd ~]$ kill 58823
[carora18@mtrx-node02pd ~]$ 
```

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.



```
[carora18@mtrx-node02pd ~]$ sleep 500
^Z
[1]+  Stopped                 sleep 500
[carora18@mtrx-node02pd ~]$ ps
   PID TTY          TIME CMD
 53948 pts/2    00:00:00 bash
 59419 pts/2    00:00:00 sleep
 59428 pts/2    00:00:00 ps
[carora18@mtrx-node02pd ~]$ kill -9 59419
[carora18@mtrx-node02pd ~]$ ps
   PID TTY          TIME CMD
 53948 pts/2    00:00:00 bash
 59452 pts/2    00:00:00 ps
[1]+  Killed                  sleep 500
[carora18@mtrx-node02pd ~]$ 
```

5. What difference you noticed in SIGTERM and SIGKILL singals?

Ans: The difference between SIGTERM and SIGKILL signals is that:

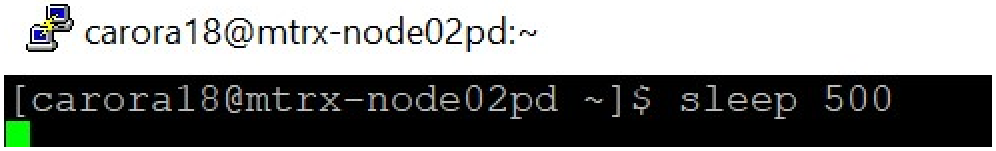
1) SIGTERM gracefully kills the process whereas SIGKILL kills the process immediately.
2) SIGTERM signal can be handled, ignored, and blocked, but SIGKILL cannot be handled or blocked.
3) SIGTERM doesn't kill the child processes. SIGKILL kills the child processes as well.

Activity 4: Foreground and background processes

1. Again ssue the following command:

   ```
   sleep 500
   ```

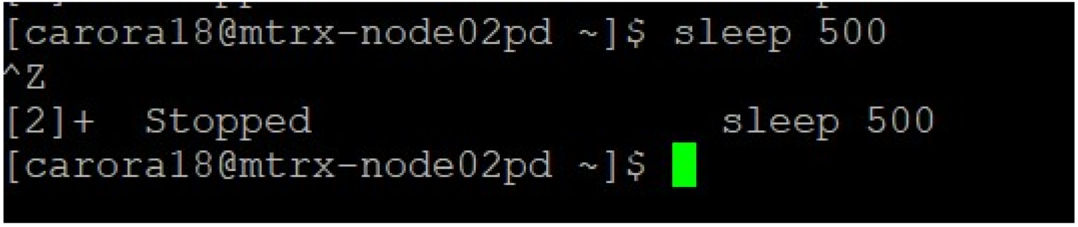
   The Unix/Linux system is designed to allow users to send **preemptive signals** to manage those processes.

   carora18@mtrx-node02pd:~

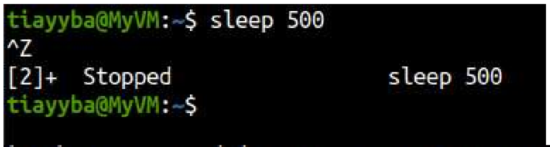
   ```
   [carora18@mtrx-node02pd ~]$ sleep 500
   ```

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

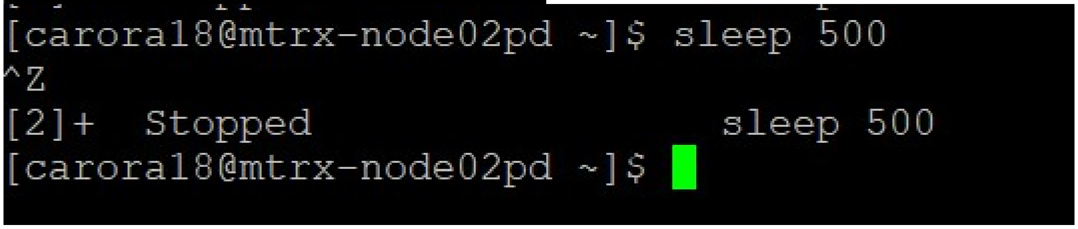
   

   ```
   [carora18@mtrx-node02pd ~]$ sleep 500
   ^Z
   [2]+  Stopped                 sleep 500
   [carora18@mtrx-node02pd ~]$
   ```

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

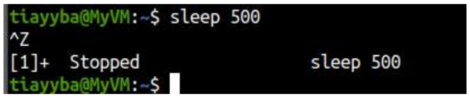
   ```
   tiayyba@MyVM:~$ sleep 500
   ^Z
   [2]+  Stopped                 sleep 500
   tiayyba@MyVM:~$
   ```

   ```
   [carora18@mtrx-node02pd ~]$ sleep 500
   ^Z
   [2]+  Stopped                 sleep 500
   [carora18@mtrx-node02pd ~]$
   ```

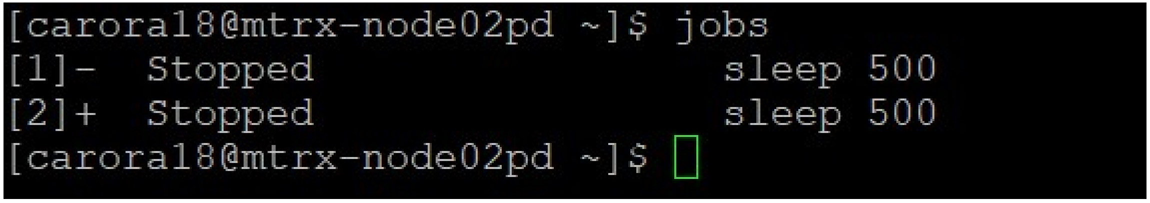
4. This indicates that this process has been placed into the **background**. This is useful in order to "**free-up**" the terminal to run other Linux commands

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



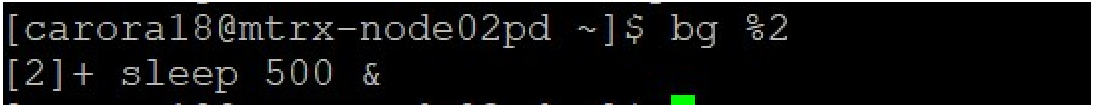
```
tiayyba@MyVM:~$ sleep 500
^Z
[1]+  Stopped                 sleep 500
tiayyba@MyVM:~$ 
```

   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.



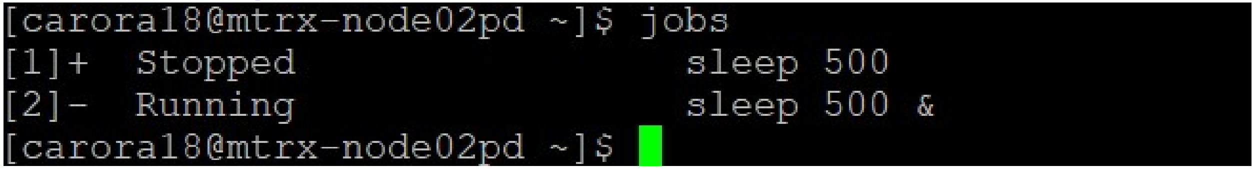
```
[caroral8@mtrx-node02pd ~]$ jobs
[1]-  Stopped                 sleep 500
[2]+  Stopped                 sleep 500
[caroral8@mtrx-node02pd ~]$ 
```

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**



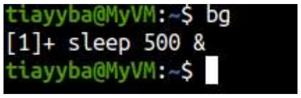
```
[caroral8@mtrx-node02pd ~]$ bg %2
[2]+ sleep 500 &
```

9. Issue the command: **jobs**



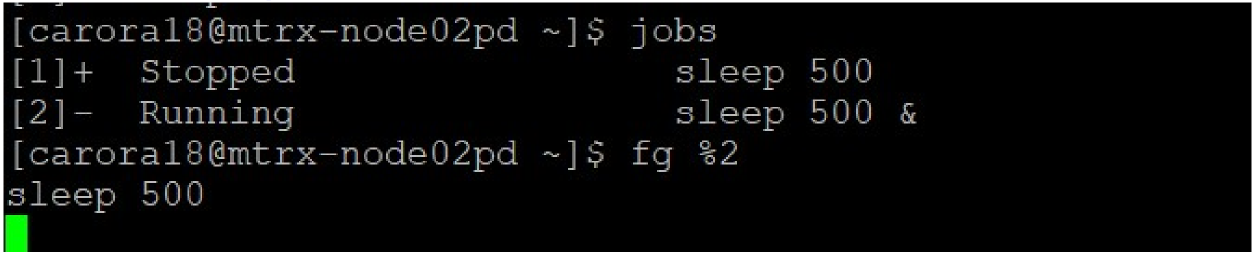
```
[caroral8@mtrx-node02pd ~]$ jobs
[1]+  Stopped                 sleep 500
[2]-  Running                 sleep 500 &
[caroral8@mtrx-node02pd ~]$ 
```

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.

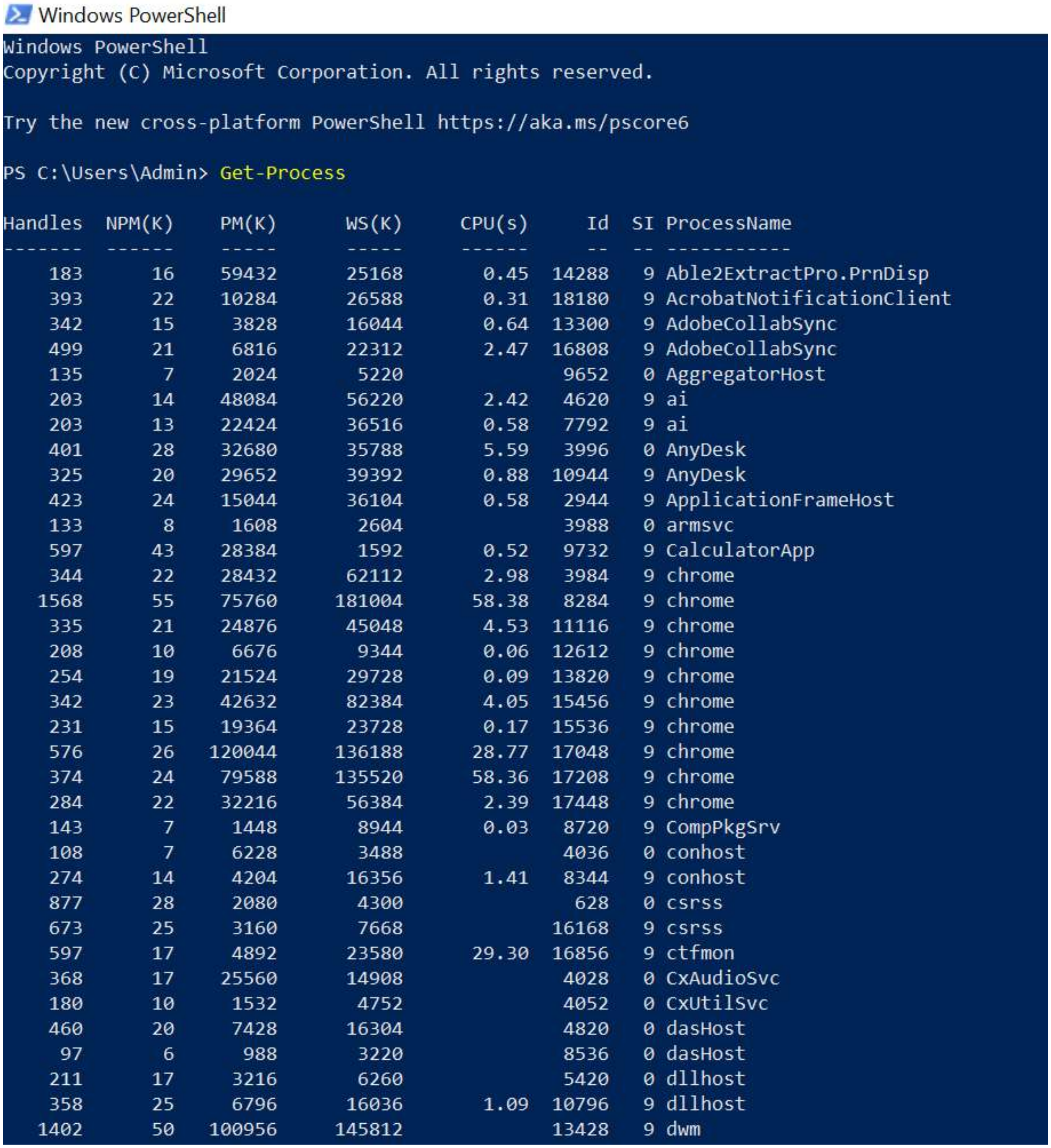


```
[caroral8@mtrx-node02pd ~]$ jobs
[1]+  Stopped                 sleep 500
[2]-  Running                 sleep 500 &
[caroral8@mtrx-node02pd ~]$ fg %2
sleep 500
```

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



Windows PowerShell

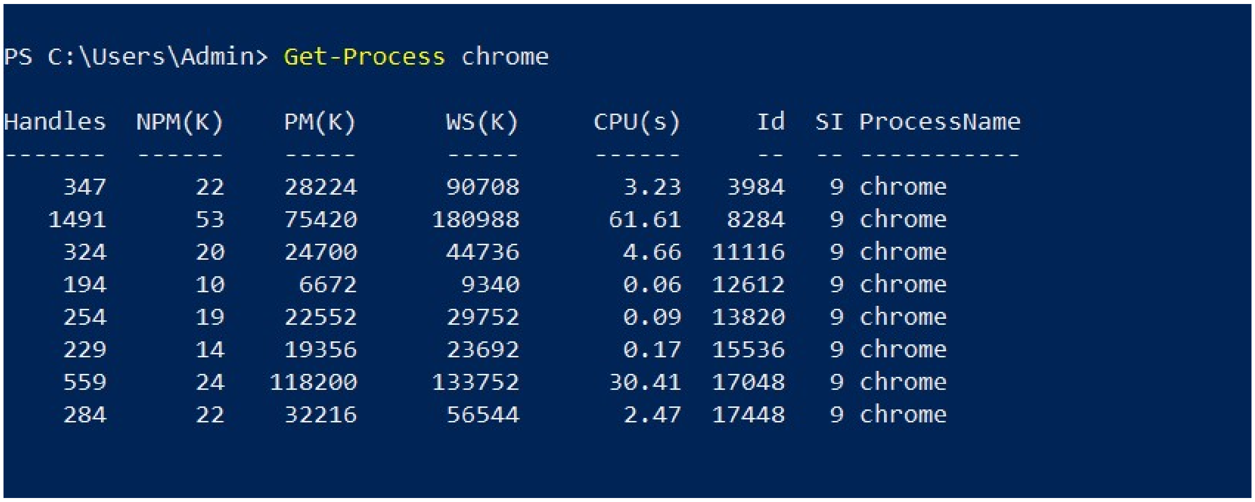
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Admin> Get-Process

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    183      16    59432      25168       0.45  14288   9 Able2ExtractPro.PrnDisp
    393      22    10284      26588       0.31  18180   9 AcrobatNotificationClient
    342      15     3828      16044       0.64  13300   9 AdobeCollabSync
    499      21     6816      22312       2.47  16808   9 AdobeCollabSync
    135       7     2024       5220              9652   0 AggregatorHost
    203      14    48084      56220       2.42   4620   9 ai
    203      13    22424      36516       0.58   7792   9 ai
    401      28    32680      35788       5.59   3996   0 AnyDesk
    325      20    29652      39392       0.88  10944   9 AnyDesk
    423      24    15044      36104       0.58   2944   9 ApplicationFrameHost
    133       8     1608       2604              3988   0 armsvc
    597      43    28384       1592       0.52   9732   9 CalculatorApp
    344      22    28432      62112       2.98   3984   9 chrome
   1568      55    75760     181004      58.38   8284   9 chrome
    335      21    24876      45048       4.53  11116   9 chrome
    208      10     6676       9344       0.06  12612   9 chrome
    254      19    21524      29728       0.09  13820   9 chrome
    342      23    42632      82384       4.05  15456   9 chrome
    231      15    19364      23728       0.17  15536   9 chrome
    576      26   120044     136188      28.77  17048   9 chrome
    374      24    79588     135520      58.36  17208   9 chrome
    284      22    32216      56384       2.39  17448   9 chrome
    143       7     1448       8944       0.03   8720   9 CompPkgSrv
    108       7     6228       3488              4036   0 conhost
    274      14     4204      16356       1.41   8344   9 conhost
    877      28     2080       4300               628   0 csrss
    673      25     3160       7668             16168   9 csrss
    597      17     4892      23580      29.30  16856   9 ctfmon
    368      17    25560      14908              4028   0 CxAudioSvc
    180      10     1532       4752              4052   0 CxUtilSvc
    460      20     7428      16304              4820   0 dasHost
     97       6      988       3220              8536   0 dasHost
    211      17     3216       6260              5420   0 dllhost
    358      25     6796      16036       1.09  10796   9 dllhost
   1402      50   100956     145812             13428   9 dwm
```

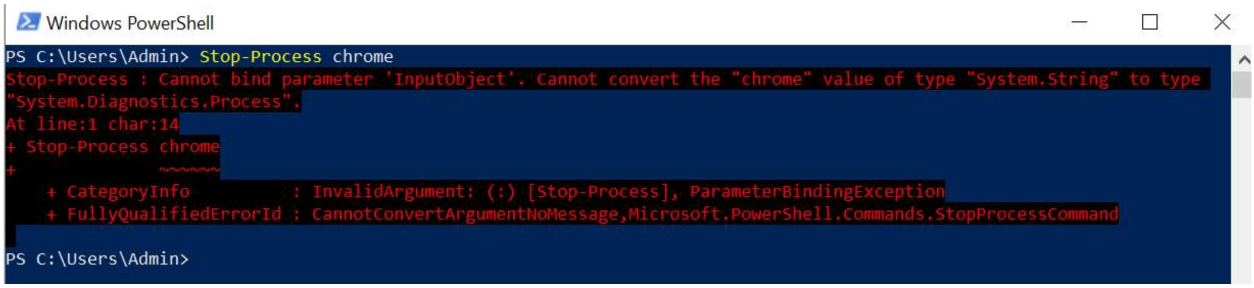
2. Explain the meaning of column headers of the information output by this command
   The column headers in the information output by the command are:
   Ans:-
   Handles:- The number of handles that the process has opened.
   NPM(K):- Non-Paged Memory in Kilobytes, which is memory that cannot be swapped out to disk and must stay in physical memory.
   PM(K):- Paged Memory in Kilobytes, which is used when a computer runs out of physical memory and begins to swap data between disk and RAM.
   WS(K):- Working Set Size in Kilobytes, which is the amount of memory used by a process that can be shared among other processes or is unique to it.
   VM(M):- Virtual Memory size in Megabytes, indicating how much virtual memory is reserved for the process.
   CPU(s):- The total processor time, in seconds, that the process has used since it started.
   Id:- Process ID number assigned by the operating system to identify each running process uniquely.
   ProcessName:- The name of the running process.
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 show the screenshot below.
   Ans:- I don't have firefox.So, I use chrome instead of firefox.



```
PS C:\Users\Admin> Get-Process chrome

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    347      22    28224      90708       3.23   3984   9 chrome
   1491      53    75420     180988      61.61   8284   9 chrome
    324      20    24700      44736       4.66  11116   9 chrome
    194      10     6672       9340       0.06  12612   9 chrome
    254      19    22552      29752       0.09  13820   9 chrome
    229      14    19356      23692       0.17  15536   9 chrome
    559      24   118200     133752      30.41  17048   9 chrome
    284      22    32216      56544       2.47  17448   9 chrome
```

4. Using this same command describe with example how you can get information about multiple processes
   Ans:- Get-Process chrome would display information about all Chrome processes running on your computer.
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



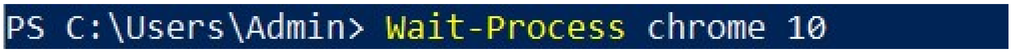
Windows PowerShell

```
PS C:\Users\Admin> Stop-Process chrome
Stop-Process : Cannot bind parameter 'InputObject'. Cannot convert the "chrome" value of type "System.String" to type
"System.Diagnostics.Process".
At line:1 char:14
+ Stop-Process chrome
+              ~~~~~~
    + CategoryInfo          : InvalidArgument: (:) [Stop-Process], ParameterBindingException
    + FullyQualifiedErrorId : CannotConvertArgumentNoMessage,Microsoft.PowerShell.Commands.StopProcessCommand

PS C:\Users\Admin>
```

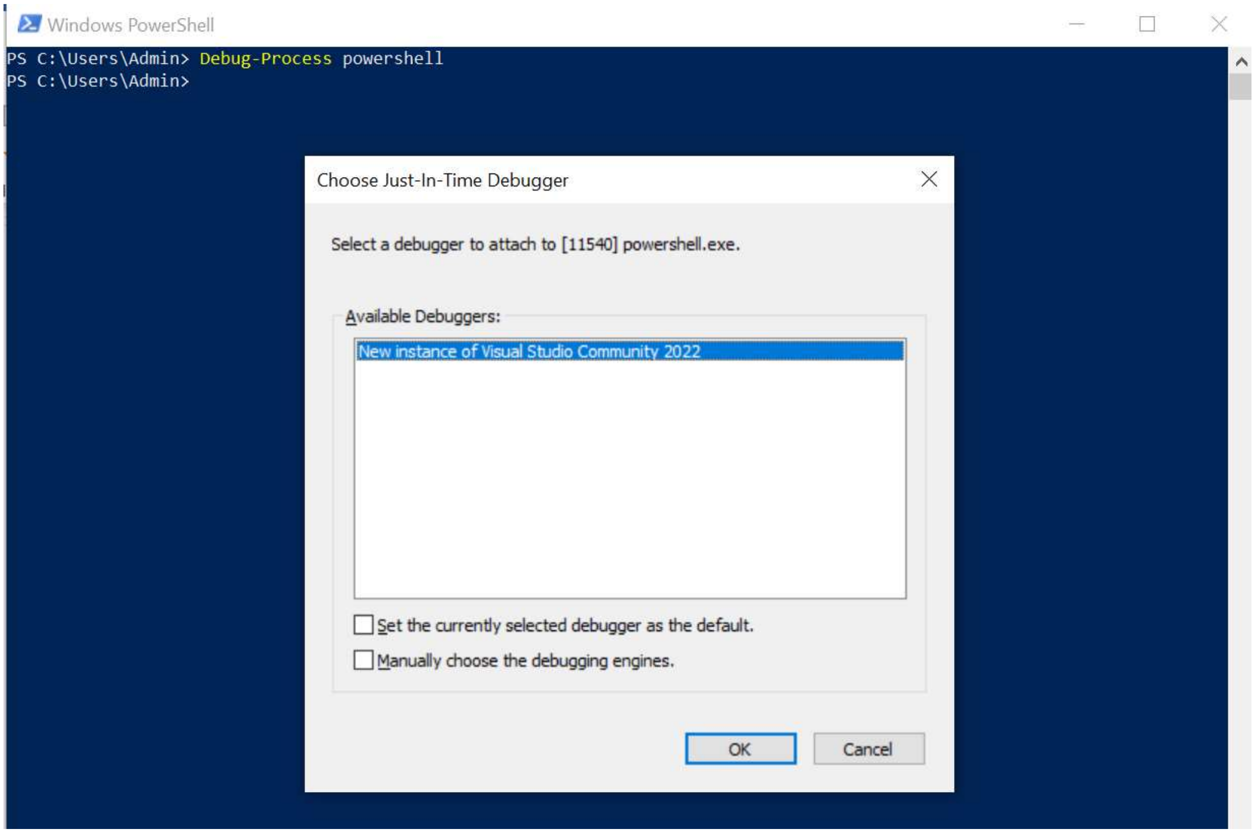
6. There are two other commands of this class to manage processes. These commands are **`Wait-Process`** and **`Debug-Process`**. Search and read about these commands and provide examples.
7.

Ans: Wait-Process: This command waits for one or more running processes to be stopped before accepting input. We can specify the processes by their name or ID, or pipe a process object to this command. We can also use the -Timeout parameter to set a maximum time for waiting. For example,



```
PS C:\Users\Admin> Wait-Process chrome 10
```

Debug-Process: This command attaches a debugger to one or more running processes on a local computer. You can specify the processes by their name or ID, or pipe a process object to this command. This command attaches the debugger that is currently registered for the process. Before using this command, you need to verify that a debugger is downloaded and configured. For example,



Windows PowerShell

```
PS C:\Users\Admin> Debug-Process powershell
PS C:\Users\Admin>
```

Choose Just-In-Time Debugger

Select a debugger to attach to [11540] powershell.exe.

Available Debuggers:

New instance of Visual Studio Community 2022

□ Set the currently selected debugger as the default.

□ Manually choose the debugging engines.

Further Practice Questions.

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

1. What is a process in Linux? Answer:

   Ans: In Linux, a process is the execution of a program. Each process has its own unique process ID (PID) and is allocated its own memory space.

2. Name three different states a process can be in, and briefly describe each state.

   a) State 1: Running

   Description: The technique is presently being done by the CPU. It is actively the usage of CPU assets to perform its duties.

   b) State 2: Sleeping

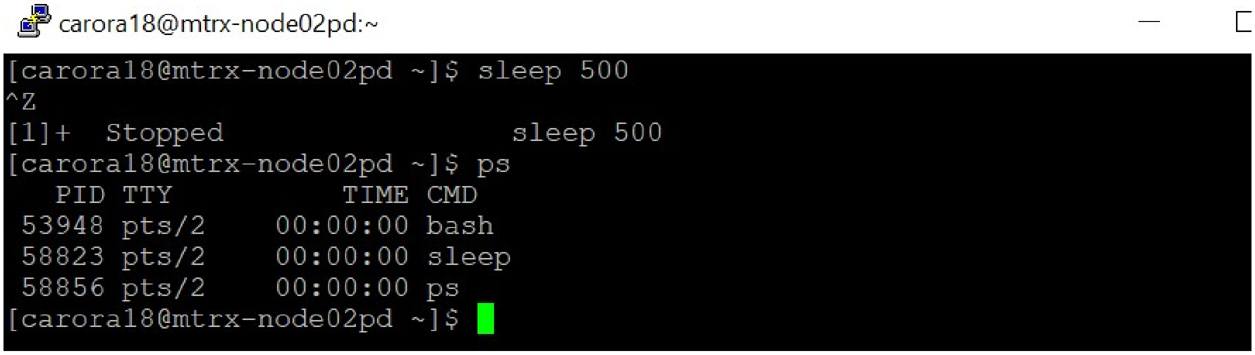
   Description: The technique is anticipating a few occasion to occur, along with I/O crowning glory, before it is able to retain executing. While on this country, the process isn't always the usage of CPU assets and is normally anticipating external input.

   c) State 3: Stopped

   Description: The procedure has been stopped, both via a consumer or by using a sign. It is no longer executing and is not scheduled to run until it's miles explicitly resumed. Processes on this nation do not eat CPU sources.

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

   Example:



```
[carora18@mtrx-node02pd ~]$ sleep 500
^Z
[1]+  Stopped                 sleep 500
[carora18@mtrx-node02pd ~]$ ps
   PID TTY          TIME CMD
 53948 pts/2    00:00:00 bash
 58823 pts/2    00:00:00 sleep
 58856 pts/2    00:00:00 ps
[carora18@mtrx-node02pd ~]$
```

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

   Ans:-

   a) PID: (Process ID) A unique number that identifies the process

   b) CPU%: This column represents CPU usage, indicating how much CPU time (in percentage) the process is consuming. This is calculated by dividing the CPU time used by the process by the total CPU time available in the system.

   c) MEM%: This column indicates how much memory (in percentage) from RAM is being used by the process. This is calculated by dividing the resident size of the process by the total physical memory available in the system.

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

   Ans:-

   Method 1: SIGTERM gracefully kills the process.

   Method 2: SIGKILL kills the process immediately.

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

   Ans:-

   Purpose of top: The top command in Linux is used to provide dynamic, real-time information about running processes, system resource usage, and other system activities.

   Sorting processes in top: Press M to sort processes by memory usage.

   Press P to revert to the default sorting by CPU usage.

   Press N to sort processes numerically by PID.

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

   Ans:- It is important to use warnings when completing procedures in Linux because ending the process abruptly will have unintended consequences and will certainly upset the balance of the gadget or lose objective facts.

Data loss: Quitting a process without permission to release current obligations can also result in data loss or corruption.

System Stability: Certain strategies are required to ensure proper functioning of the device. Suddenly stopping those processes can cause system instability, corruption, or unpredictability.

Storage: Processors typically manage resources by allocating file handles, network connections, or memory. When one path ends up nicely freeing up those sources, it is capable of leaking useful features, ultimately affecting both system performance and stability

Dependencies: There may be different options depending on different methods or even devices provided. Removing one system at once can further compromise the capabilities of an array of channels or projects, creating a spiraling curve.

Implementation Impact: Carefully removing the process and relying on those processes can impact users or businesses, causing disruption or delays stop working fruit.

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

Ans:-

The kill and killall commands in Linux are both used to terminate processes, but they differ in how they identify the processes to be terminated:

kill:

kill is used to terminate processes based on their Process ID (PID). We specify the PID of the process you want to terminate as an argument to the kill command.

It sends a signal to the specified process, instructing it to terminate. By default, it sends the TERM signal, allowing the process to gracefully exit. However, we can specify different signals, such as SIGKILL (-9), to force termination.

killall:

killall is used to terminate processes based on their name rather than their PID. We specify the name of the process we want to terminate as an argument to the killall command.

It sends signals to all processes with matching names, instructing them to terminate. As with kill, we can specify different signals if needed.

9. True or False: Terminating a process with SIGKILL allows it to perform cleanup operations before termination.

   Answer: False

10. Name two signals that can be sent to a process using the kill command, and briefly describe their effects.

    Signal 1: SIGTERM

    Effect: This signal requests the process to terminate gracefully. The process has the opportunity to perform cleanup operations before exiting.

    Signal 2: SIGKILL

    Effect: This signal forcefully terminates the process immediately without giving it a chance to handle the signal or perform any cleanup tasks.