**Process Management Commands**

ps

Displays the processes of the current shell session.

ps -e or ps -A

Show all processes.

ps -f

Full-format listing (provides additional details).

ps -l

Long format with detailed information.

ps -u <username>

Show processes owned by a specific user.

ps -p <pid>

Show processes with a specific PID.

ps -C <command>

Show processes based on the command name

ps aux : Shows all processes with detailed information (commonly used).

* **a**: Show processes for all users.
* **u**: Display in user-oriented format.
* **x**: Show processes without a controlling terminal.

ps -ef : Shows all processes in full-format (another common option).

* **e**: All processes.
* **f**: Full-format listing.

ps -e --format pid,cmd

Shows all processes in format of pid,cmd

ps --sort=-%mem

sorts the list of processes by memory usage in descending order.

ps -U <user> or ps -u <user>

Show processes for a user, based on user ID.

ps -t <tty>

Show processes running on a particular terminal.

**Signal Related:**

ps -o pid,comm,etime

Display specific columns like process ID, command name, and elapsed time.

**Environment Variables:**

ps e

Display the environment variables of each process.

**Thread Information:**

ps -L -p <pid>

Show threads of a specific process

**top**

Monitoring system performance and managing processes

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND

**Options for top command**

**q**: Quit top.

**h**: Display help information.

**k**: Kill a process by entering its PID.

**r**: Renice *a* process (change its priority).

**P**: Sort processes by CPU usage.

**M**: Sort processes by memory usage.

**T**: Sort processes by runtime.

**c**: Toggle between displaying the command name and the full command line

fg

**Started a job(process) in the background**

sleep 100 &

**Bring the background job to foreground**

fg %<job\_number>

bg

**We can resume or start the foreground job back to background**

bg % <job\_id>

Any processes which are running in background or foreground, if we close the terminal those processes will get killed

**nohup (No hang up)**

If we use the nohup command with any command, it keeps running even after you log out.

->nohup sleep 1000 &

(sleep command is running in the background, if we close the terminal also this process will keep running)

After login again to the terminal, we can check this process is being running

->ps -C sleep

We can set nohup command later also using disown command

->sleep 1542 & (running sleep command in background)

->jobs (Run the jobs command to know the job id & process id)

->disown <process\_id>

->disown %<job\_id>

Now sleep command run as nohup

**Kill**

Primary function is to terminate processes.

It can also send other signals that can instruct a process to pause, continue, or perform other actions.

**Terminate a process named my\_script.sh?**

->pgrep my\_script.sh (gives the pid of process my\_script.sh)

->kill -9 <pid> (kills the process my\_script.sh)

**Options:**

**-3 (SIGQUIT)**: Sends a signal to quit and create a core dump for debugging.

**-9 (SIGKILL)**: Forces termination of a process without cleanup.

**-18 (SIGCONT)**: Resumes a paused process.

**-19 (SIGSTOP)**: Immediately stops a process.

**Strace**

* The strace command in Linux is a powerful diagnostic tool used to monitor and trace system calls made by a process.
* It helps in understanding how programs interact with the operating system, making it invaluable for debugging and performance analysis.

Allow you to trace system calls made by a program and the signals received by it.

->strace [options] <command>

**Options:**

**-p <pid>**: Attach to a running process with the specified PID.

**-f**: Trace child processes as they are created using fork().

**-e <expr>**: Specify an expression to filter the events to trace (e.g., -e trace=open to trace only open calls).

**-o <filename>**: Write the output to a specified file instead of standard output.

**-c**: Count time, calls, and errors for each system call and print a summary at the end.

**-tt**: Print timestamps with microsecond precision.

pkill

pkill is used to send signals to processes based on specified criteria, similar to pgrep. It can terminate processes, restart them, or send any other signal.

pkill [options] pattern

**Common Options**:

* -u <user>: Send the signal to processes owned by the specified user.
* -f: Match against the full command line.
* -n: Send the signal only to the newest process.
* -o: Send the signal only to the oldest process.
* -signal: Specify a signal to send (e.g., -9 for SIGKILL).

pkill <process name>

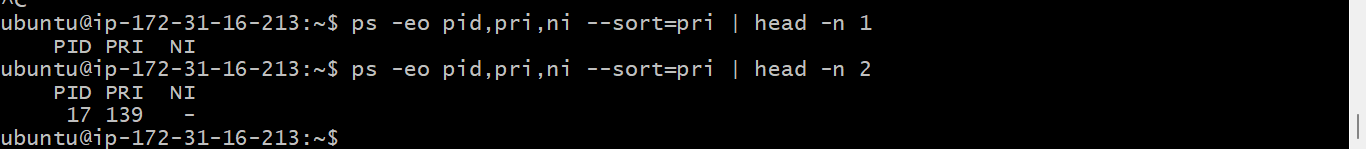
pkill -9 <process name>

pkill -u <username>

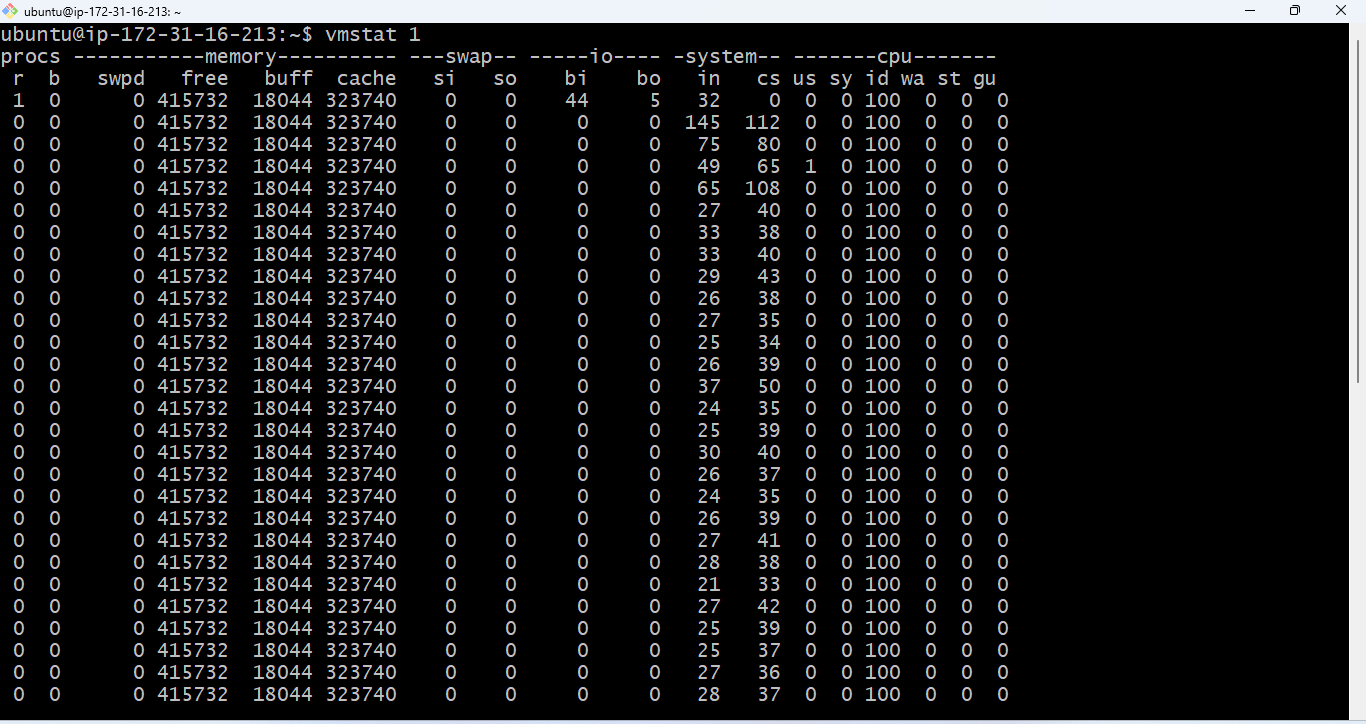
**Assignment 5 -Process Management**

1. List out highest priority process in the system

->ps –eo pid,pri,ni –sort=pri | head –n 2



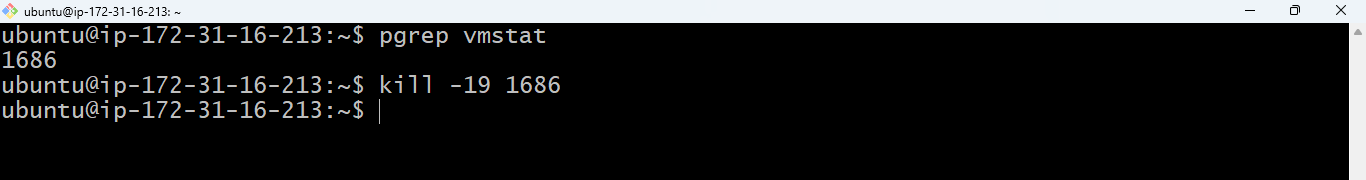
1. Open terminal with 2 tabs or sessions
2. run command "vmstat 1"

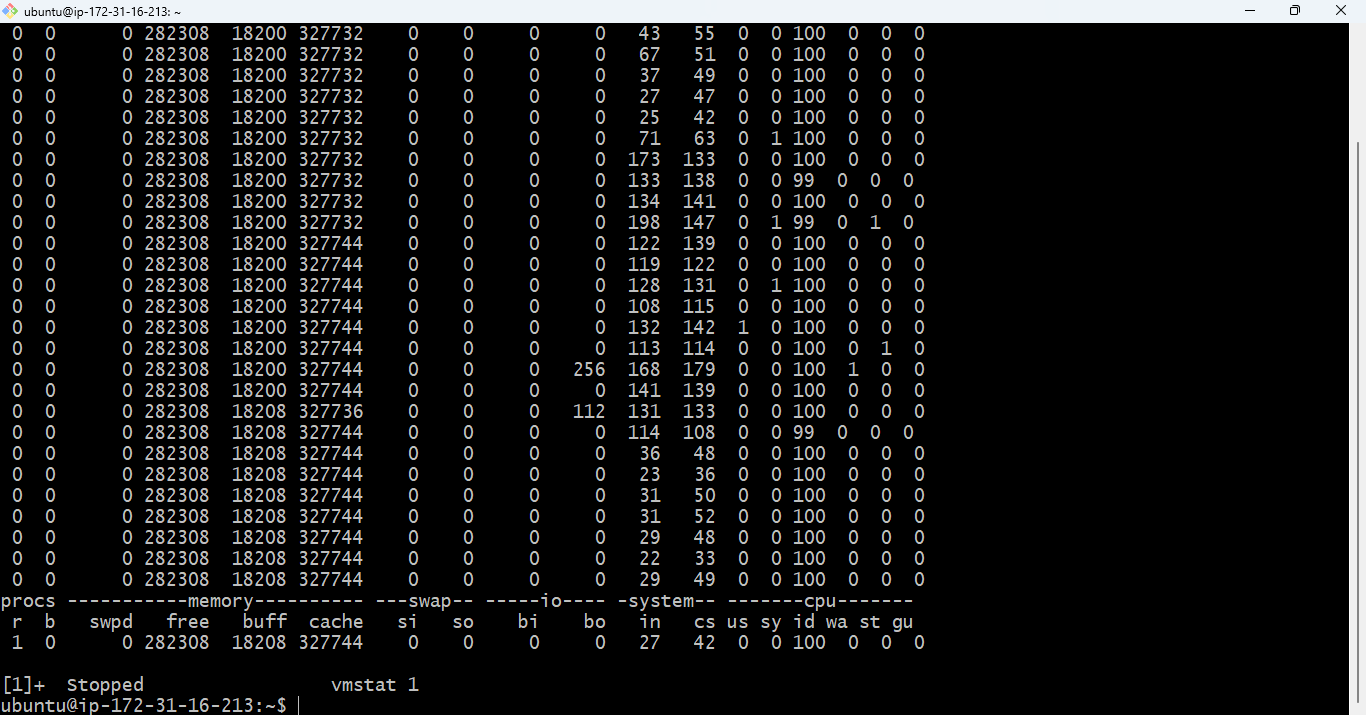


1. switch to another tab, pause running vmstat command for few seconds and resume it again use appropriate SIGNALS to do this activity.

**Pause the vmstat command using kill -19 1686**

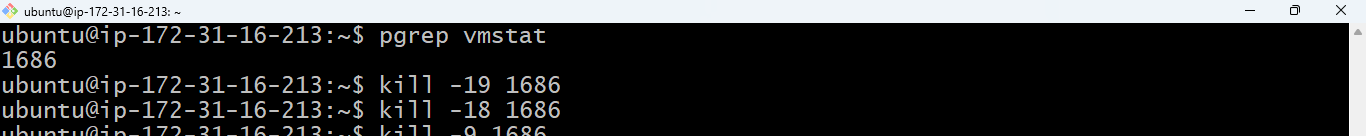
->kill -19 1686

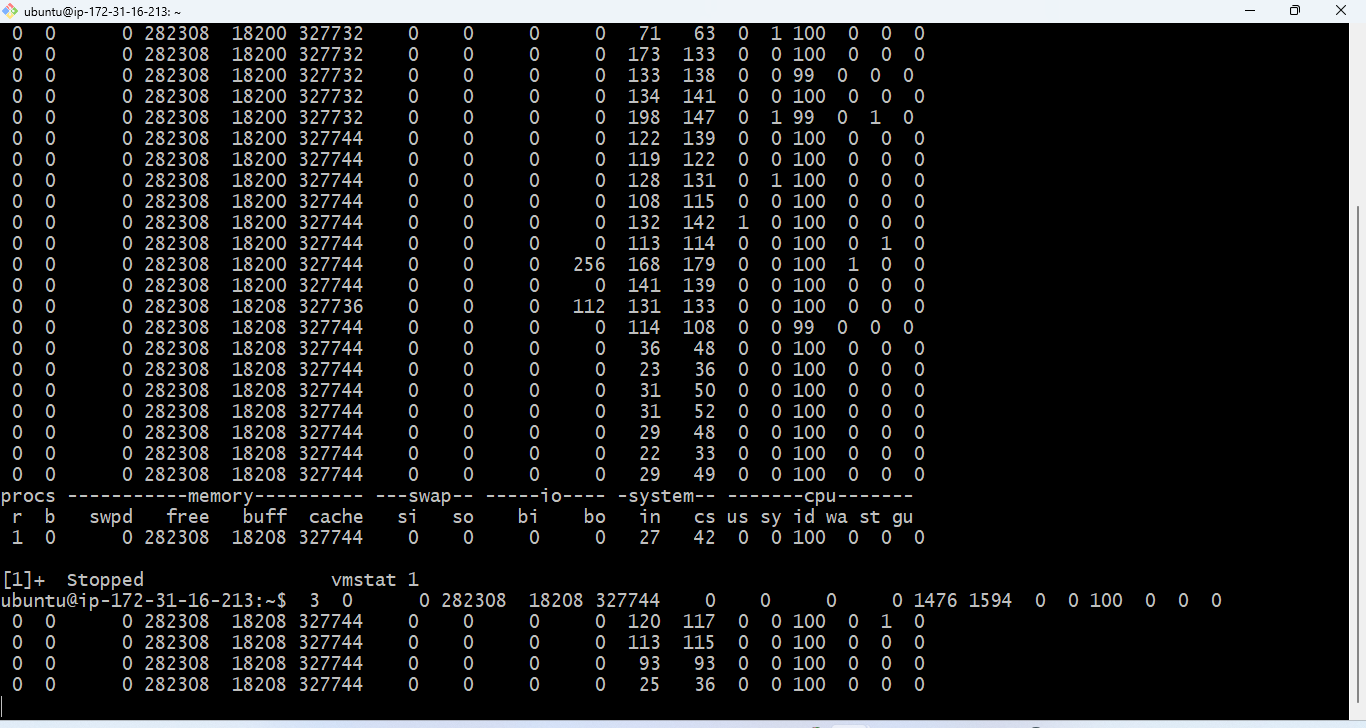


Paused /Stopped the vmstat 

**Start / Resume the vmstat using kill -18 1686**

->kill -18 1686





1. Find the process which is sleeping in "wait" state

->ps –eo pid,comm,state | grep ‘S’

