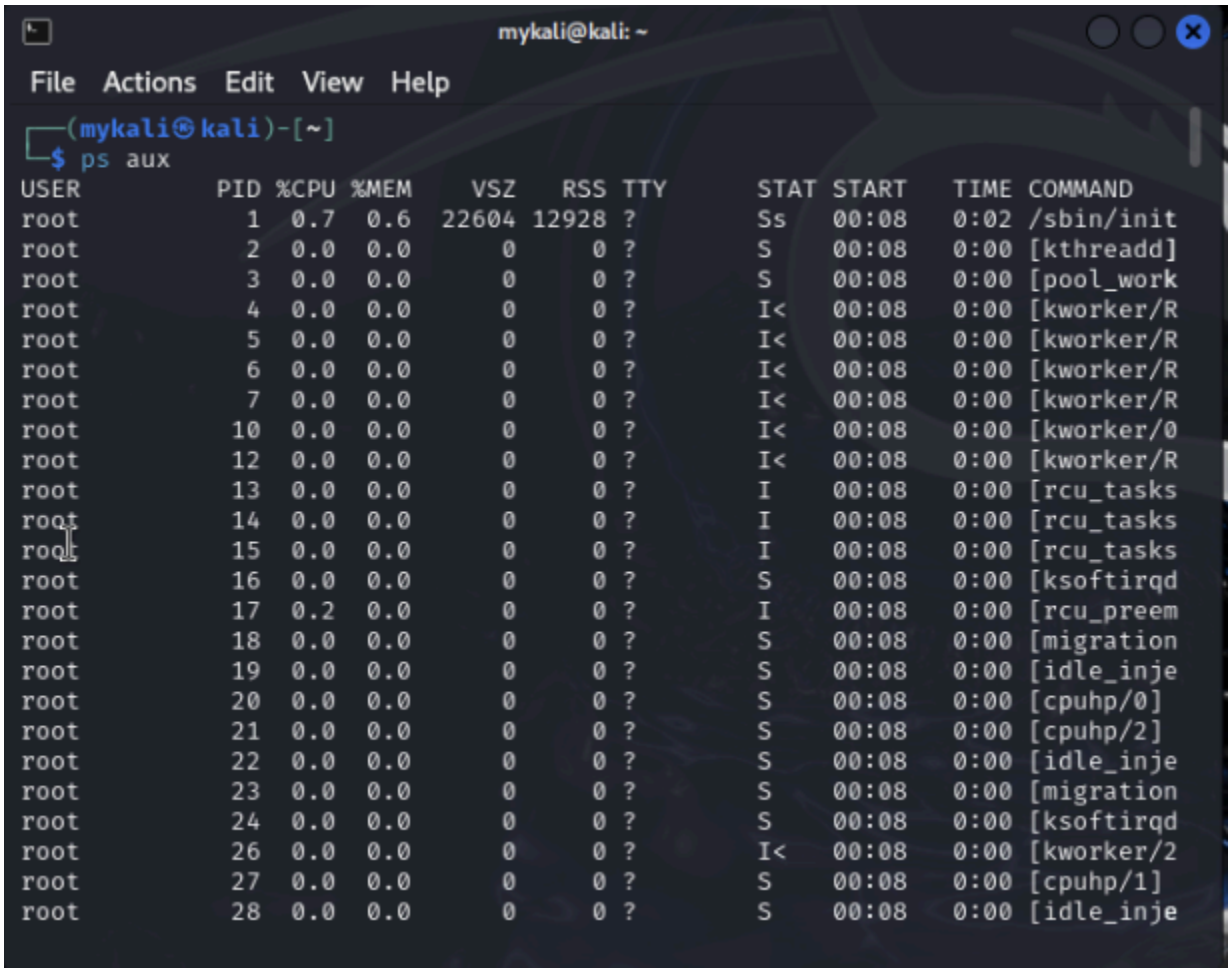
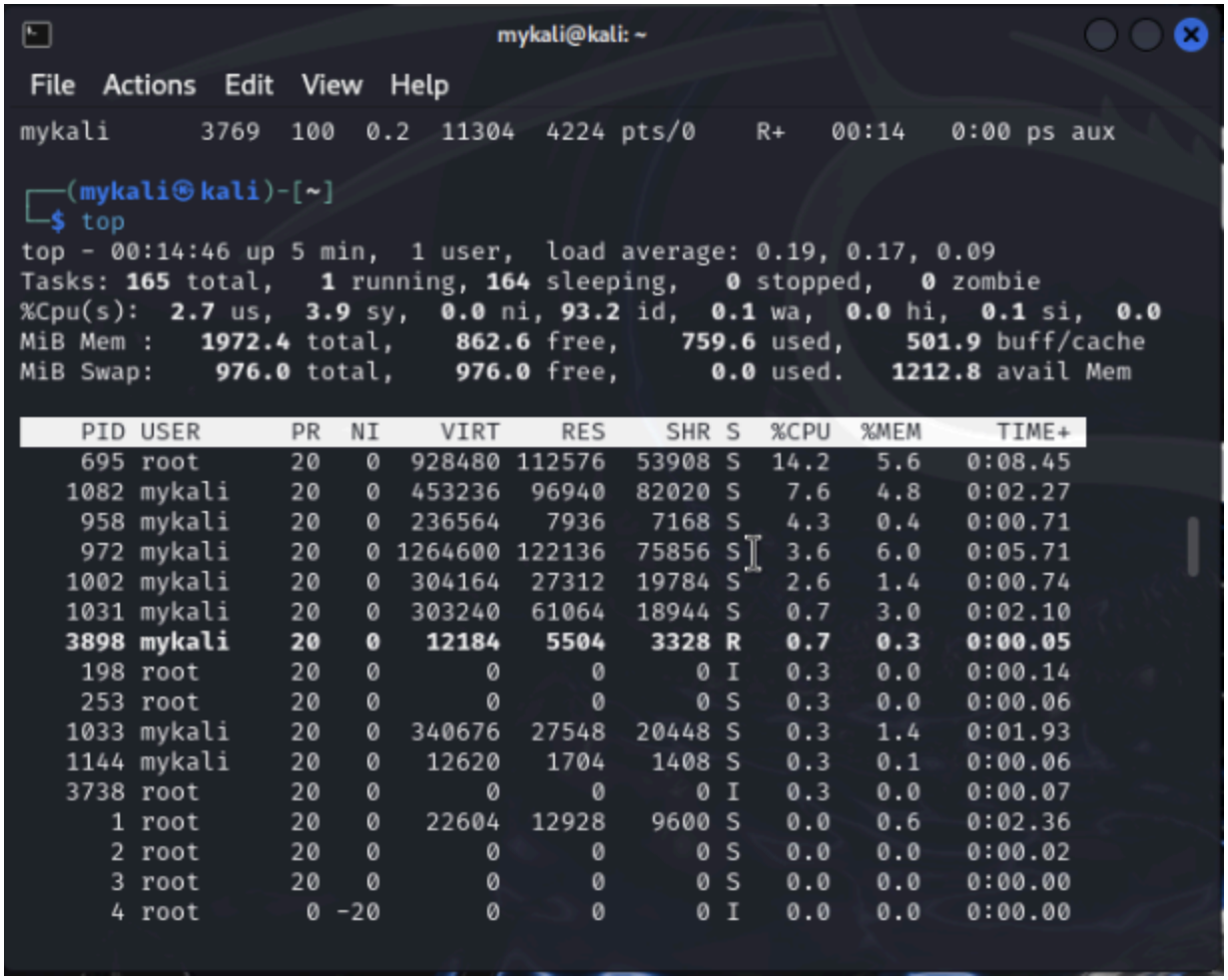
Lab 1 : Process Exploration and Identification

Task 1. List Running Processes:
Use ps, top, or htop to list all running processes on the system.
Understand the difference between ps, top, and htop, and experiment with their options (e.g., ps aux, top -u <username>).

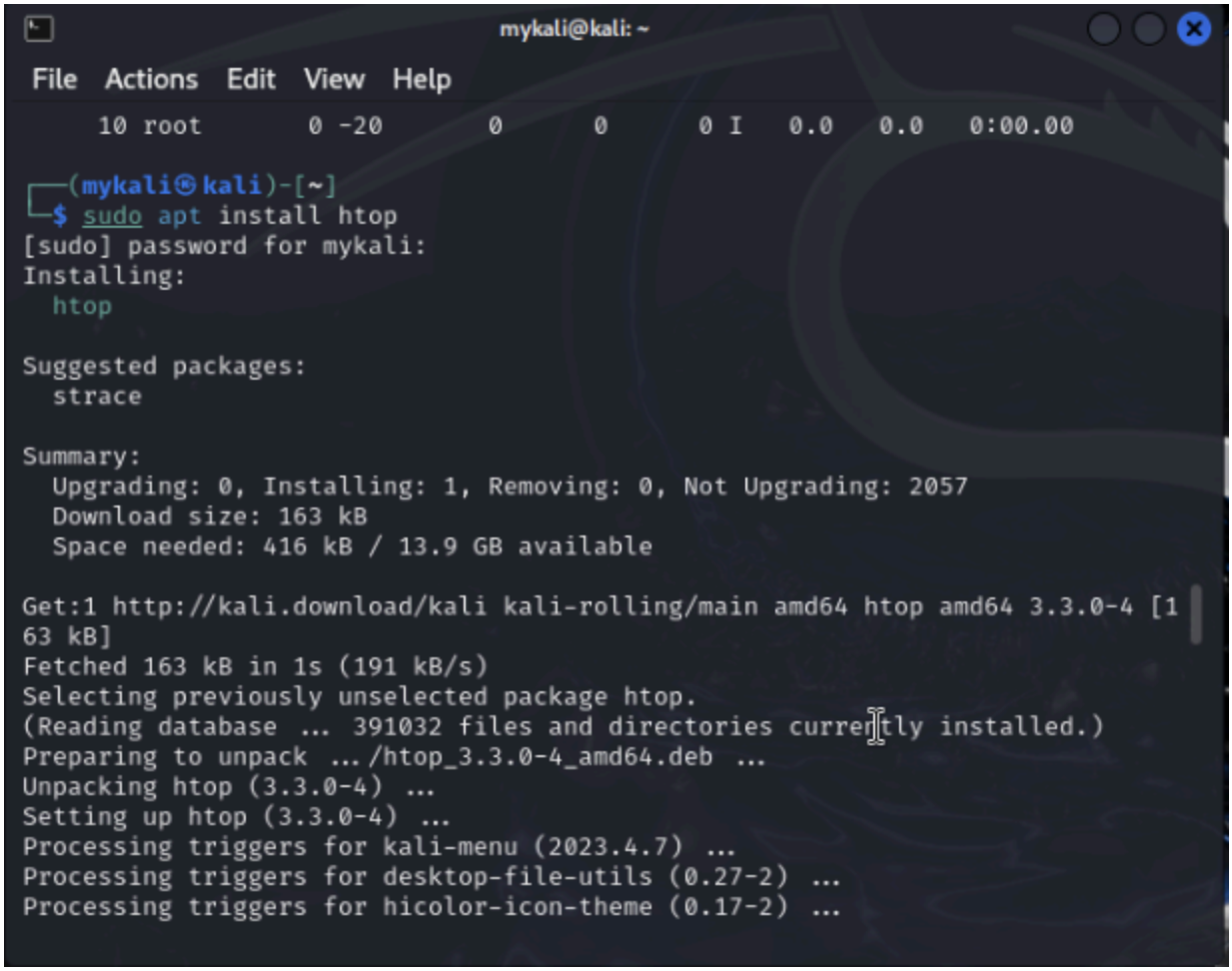
- ps aux
  NOTE: a → All users' processes
  u → User-oriented format
  x → Include processes without a terminal (daemon processes)



- top
  NOTE: Real-time updating list of processes
  Press q to exit.
  Try `top -u <username>` to filter by a specific user.



- htop
  NOTE: If not installed:
  `sudo apt install htop`
  Use arrow keys to navigate and `F10` to exit.



Task 2. Find a Specific Process:

Use pgrep to find the PID (process ID) of a specific running process like apache2 or nginx.

- pgrep -l apache2
  NOTE: flag -l is used to print PID along with process name

Use pstree to view a tree of processes and their parent-child relationships.

- pstree -p
  NOTE: Add `-u <username>` to filter for a specific user

Task 3. Investigate Process Details:

Use lsof to identify files opened by a process.

- lsof -p [PID to be mentioned here]

Check the memory usage and CPU time of a process using ps -eo pid,etime,%mem,%cpu,comm.

- ps -eo pid,etime,%mem,%cpu,comm | grep [process_name]

  NOTE: `pid` → Process ID

  `etime` → Elapsed running time

  `%mem` → Memory usage

  `%cpu` → CPU usage

  `comm` → Command name

Lab 2 : Process Control and Termination

Task 1. Send Signals to Processes:

Use kill to send signals to processes. Try sending a SIGTERM and SIGKILL to terminate a process by PID.

- apache2 & (Start apache process)
- sudo kill [PID] (Eg: kill 13593)
- sudo kill -s SIGTERM [PID]
- sudo kill -s SIGKILL [PID]

```
mykali@kali: ~
File  Actions  Edit  View  Help

┌──(mykali㉿kali)-[~]
└─$ pgrep -l apache2
4373 apache2
4376 apache2
4377 apache2
4378 apache2
4379 apache2
4380 apache2

┌──(mykali㉿kali)-[~]
└─$ kill 4373
kill: kill 4373 failed: operation not permitted

┌──(mykali㉿kali)-[~]
└─$ sudo kill 4373

┌──(mykali㉿kali)-[~]
└─$ sudo kill -s SIGKILL 4373
kill: (4373): No such process
```

Use kill -s STOP <PID> and kill -s CONT <PID> to stop and resume a process.

- kill -s STOP [PID]
- kill -s CONT [PID]

```
┌──(mykali㉿kali)-[~]
└─$ pgrep -l apache2
17645 apache2
17648 apache2
17649 apache2
17650 apache2
17651 apache2
17652 apache2

┌──(mykali㉿kali)-[~]
└─$ sudo kill -s STOP 17645

┌──(mykali㉿kali)-[~]
└─$ sudo kill -s CONT 17645

┌──(mykali㉿kali)-[~]
└─$
```

Task 2. Send Custom Signals:

Send a SIGINT signal to a running process (e.g., when running a program in the terminal, use Ctrl+C or kill -s SIGINT <PID>).

- kill -s SIGINT [PID]

Task 3. Test Process Termination:
Start a process, for example, sleep 300, then find its PID and try to terminate it using kill or kill -9.

- sleep 300 &
- pgrep sleep
- kill -9 [PID]
- pgrep sleep (To verify if the process is still running)

-

Lab 3 : Managing Background and Foreground Processes

Task 1. Run a Process in the Background:
Start a process in the background using &, e.g., sleep 100 &.

- sleep 100 &

Use jobs to see a list of background jobs.

- Jobs
-

Task 2. Bring a Process to the Foreground:

Use the fg command to bring a background process to the foreground.

- fg %[JOB NUMBER]

```
┌──(mykali㉿kali)-[~]
└─$ fg %1
[1]  + running    sleep 100
```

Task 3. Pause and Resume a Process:

Pause a background process using Ctrl+Z and resume it in the background with the bg command.

- Run a command normally (e.g., `ping google.com`)
- press **`Ctrl + Z`** to pause it.
- jobs
  NOTE: job will be shown as stopped
  - bg %[JOB NUMBER] (Resume job in background)
  - fg %[JOB NUMBER] (Resume job in foreground)

```
mykali@kali: ~
File  Actions  Edit  View  Help

┌──(mykali㉿kali)-[~]
└─$ ping google.com
PING google.com (142.250.183.110) 56(84) bytes of data.
64 bytes from bom12s13-in-f14.1e100.net (142.250.183.110): icmp_seq=1 ttl=110
 time=42.6 ms
64 bytes from bom12s13-in-f14.1e100.net (142.250.183.110): icmp_seq=2 ttl=110
 time=41.1 ms
64 bytes from bom12s13-in-f14.1e100.net (142.250.183.110): icmp_seq=3 ttl=110
 time=43.0 ms
^Z
zsh: suspended  ping google.com

┌──(mykali㉿kali)-[~]
└─$ jobs
[1]  + suspended  ping google.com

┌──(mykali㉿kali)-[~]
```

Task 4. Control Multiple Jobs:

Start multiple jobs in the background and manage them with jobs, fg, and bg.

- sleep 200 &
- ping -c 5 google.com &
  NOTE: We just ran multiple commands
- jobs (Shows list of all background jobs)
- bg %[JOB NUMBER] (Resume job in background)

- fg %[JOB NUMBER] (Forward job to foreground)
- Kill %[JOB NUMBER] (Kill a job)

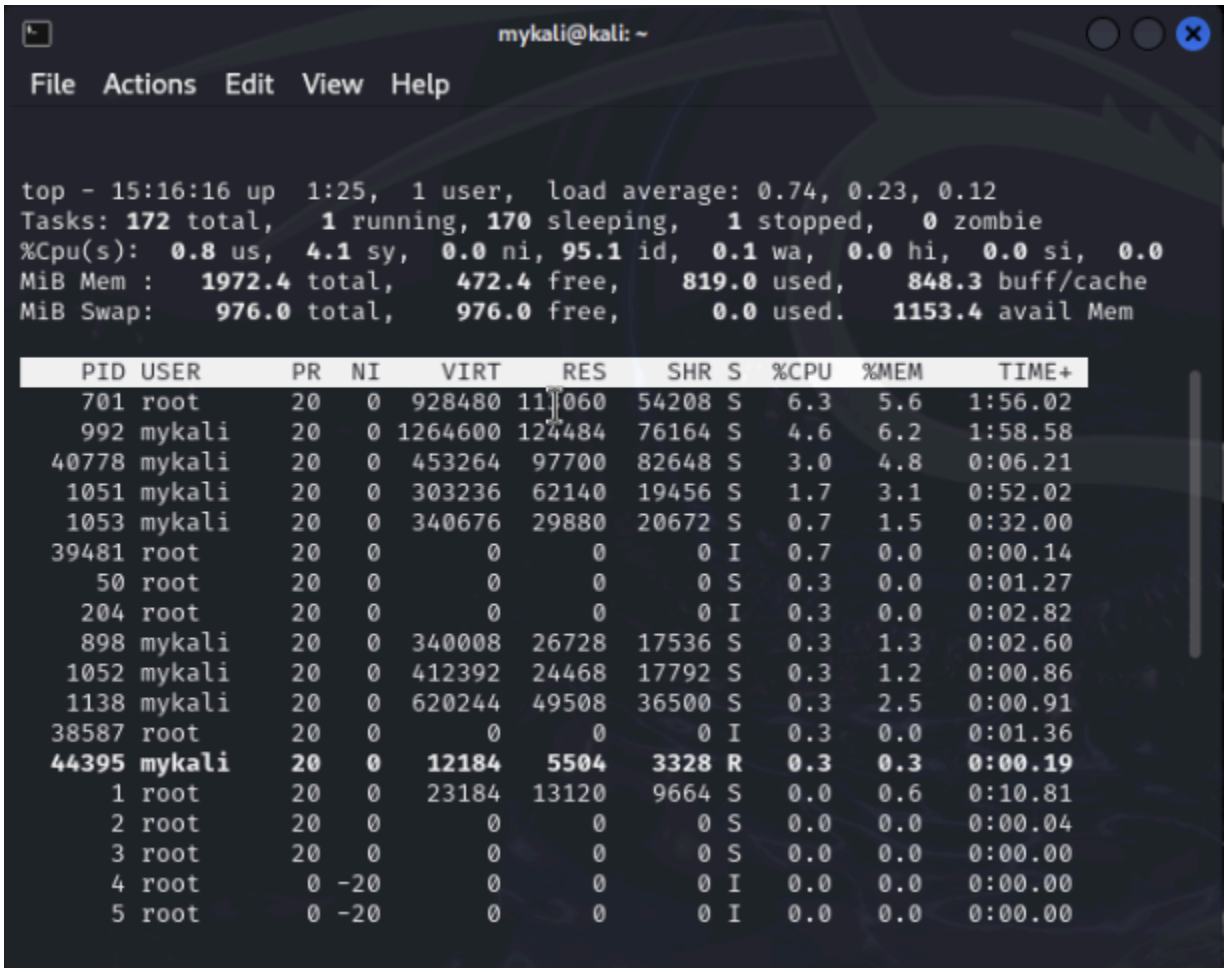
Lab 4 : Monitoring System Performance and Resource Usage

Task 1. Monitor CPU Usage:

Use top or htop to monitor CPU usage in real-time.

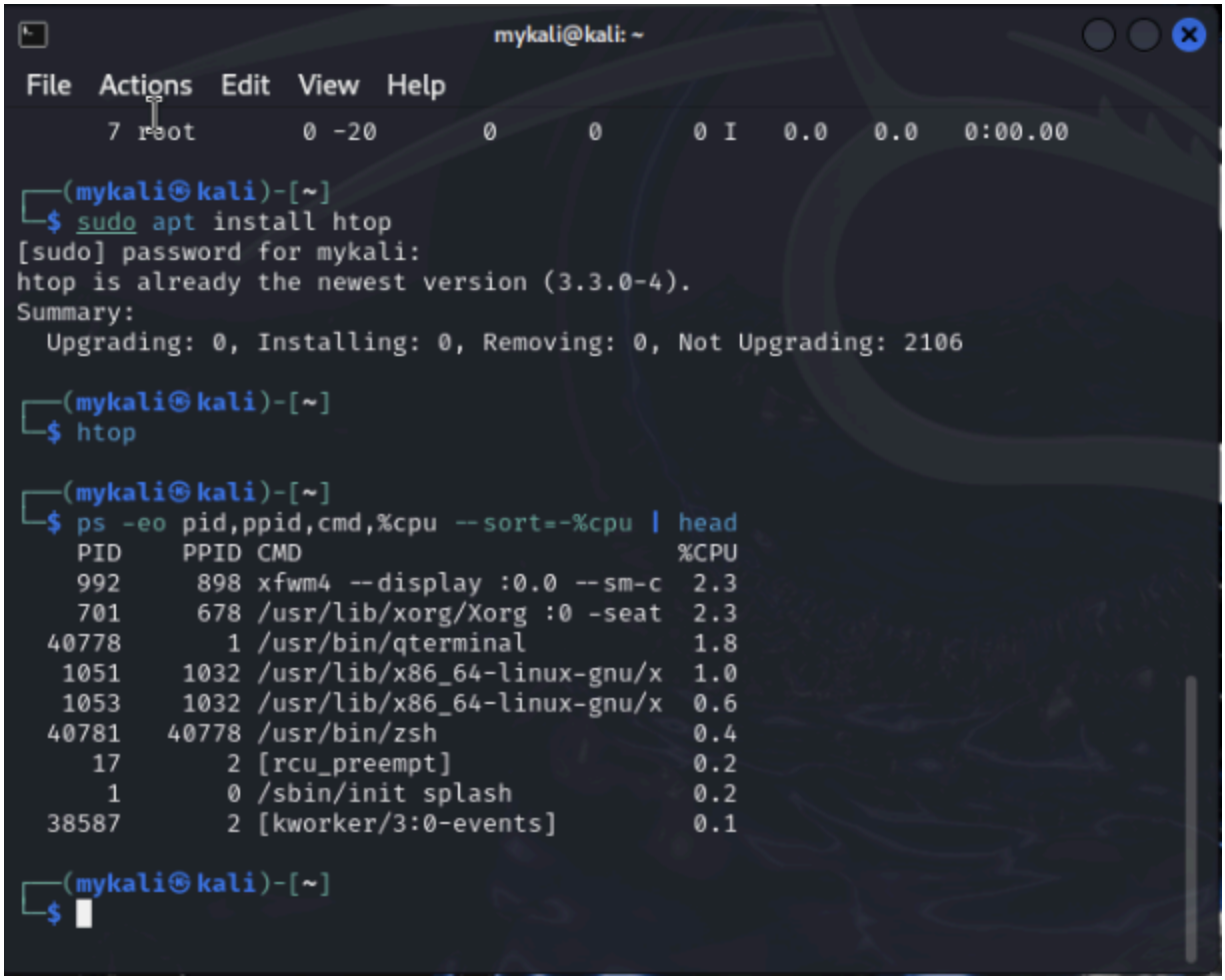
- top

  NOTE: Within `top`, press `Shift + P` to sort processes by CPU consumption.



- htop

  NOTE: If not installed:
  - sudo apt install htop



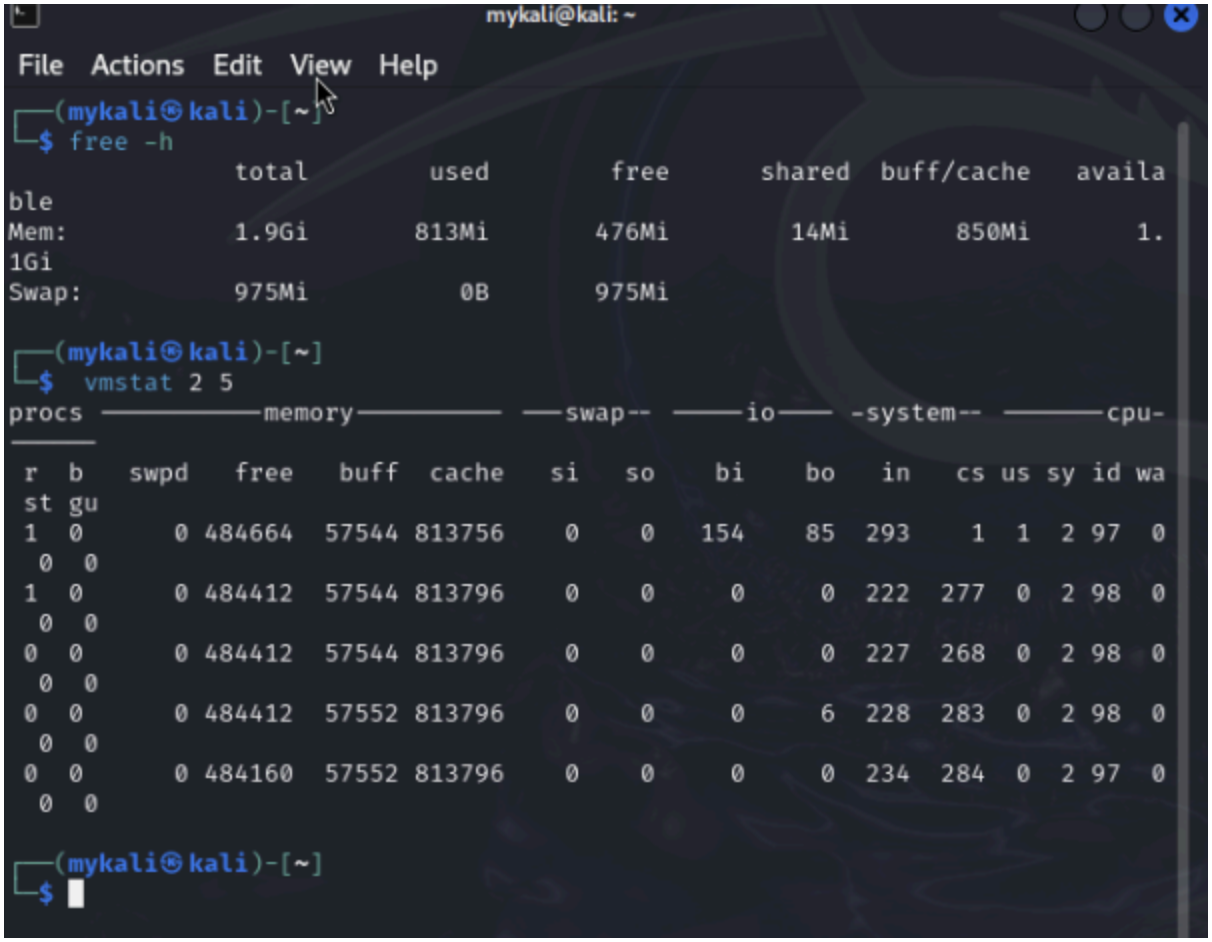
Look for processes consuming high CPU and analyze them.

- ps -eo pid,ppid,cmd,%cpu --sort=-%cpu | head

Task 2. Monitor Memory Usage:

Use free or vmstat to check system memory usage.

- free -h
- vmstat 2 5
- NOTE:
  This provides memory, CPU, and I/O stats every 2 seconds for 5 intervals.
  Columns to observe: `si`/`so`: Swap in/out.
  `free`: Available memory.
  `buff/cache`: Buffers and cache memory usage.



Use ps aux --sort=-%mem to find processes using the most memory.

- ps aux --sort=-%mem | head

```
mykali@kali: ~
File  Actions  Edit  View  Help
  0  0

┌──(mykali㉿kali)-[~]
└─$ ps aux --sort=-%mem | head
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
mykali       992  2.4  6.1 1264600 124484 ?      Sl   13:50   2:10 xfwm4 --di
splay :0.0 --sm-client-id 2057af3a9-9282-4a79-bca4-a3ed3a93c3e8
root         701  2.4  5.5 928480 113060 tty7    Ssl+ 13:50   2:10 /usr/lib/x
org/Xorg :0 -seat seat0 -auth /var/run/lightdm/root/:0 -nolisten tcp vt7 -nov
tswitch
mykali     40778  2.2  4.8 453264 97700 ?        Sl   15:08   0:16 /usr/bin/q
terminal
mykali      1051  1.0  3.0 303236 62140 ?        Sl   13:50   0:55 /usr/lib/x
86_64-linux-gnu/xfce4/panel/wrapper-2.0 /usr/lib/x86_64-linux-gnu/xfce4/panel
/plugins/libcpugraph.so 13 16777228 cpugraph CPU Graph Graphical representati
on of the CPU load
mykali      1043  0.1  3.0 477888 60672 ?        Sl   13:50   0:07 xfdesktop
--display :0.0 --sm-client-id 2bdfbfe08-7a49-44c6-9492-7ee03ae47dbe
mykali      1195  0.0  2.5 515580 52248 ?        Sl   13:50   0:02 /usr/bin/p
ython3 /usr/bin/blueman-applet
mykali      1044  0.0  2.4 461040 50056 ?        Sl   13:50   0:04 /usr/lib/x
86_64-linux-gnu/xfce4/panel/wrapper-2.0 /usr/lib/x86_64-linux-gnu/xfce4/panel
/plugins/libwhiskermenu.so 1 16777223 whiskermenu Whisker Menu Show a menu to
 easily access installed applications
mykali      1138  0.0  2.4 620244 49508 ?        Sl   13:50   0:01 nm-applet
mykali      1032  0.0  2.3 461832 47672 ?        Sl   13:50   0:03 xfce4-pane
l --display :0.0 --sm-client-id 2848635dc-3246-480c-871a-114f5b02f31d
```

Task 3. Disk Usage and I/O Monitoring:

Use iotop or dstat to monitor real-time disk I/O usage by processes.

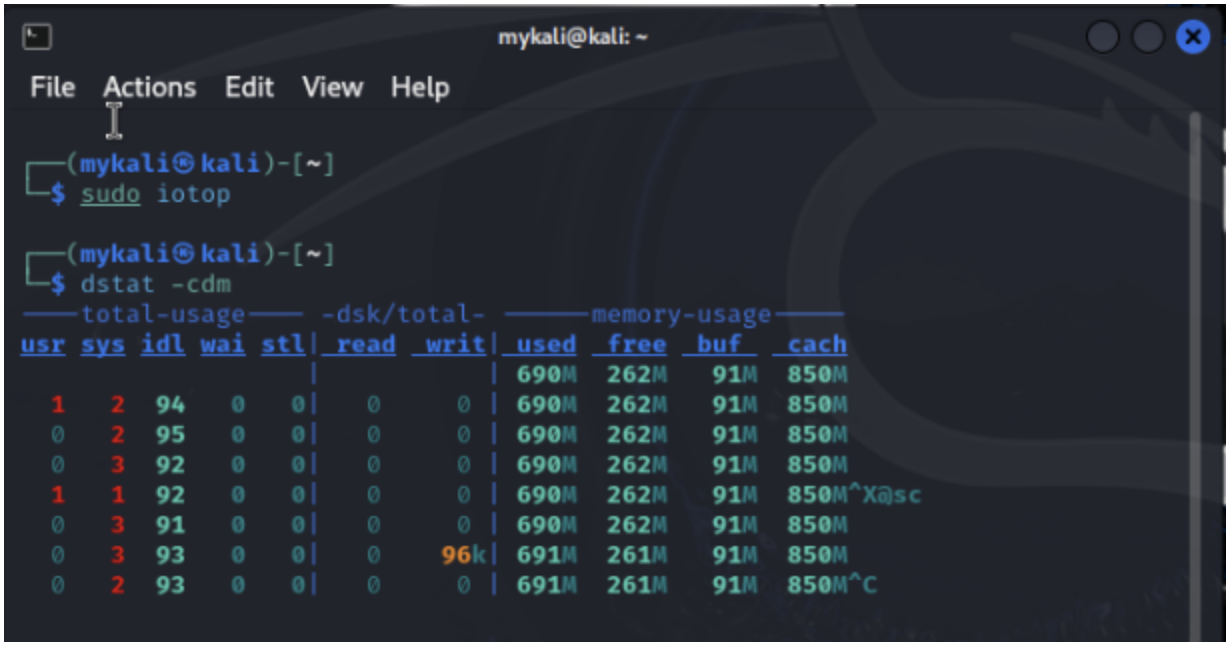
- sudo iotop

  NOTE: If needed to install:

  - Sudo apt install iotop
- dstat -cdm

NOTE: If needed tp install

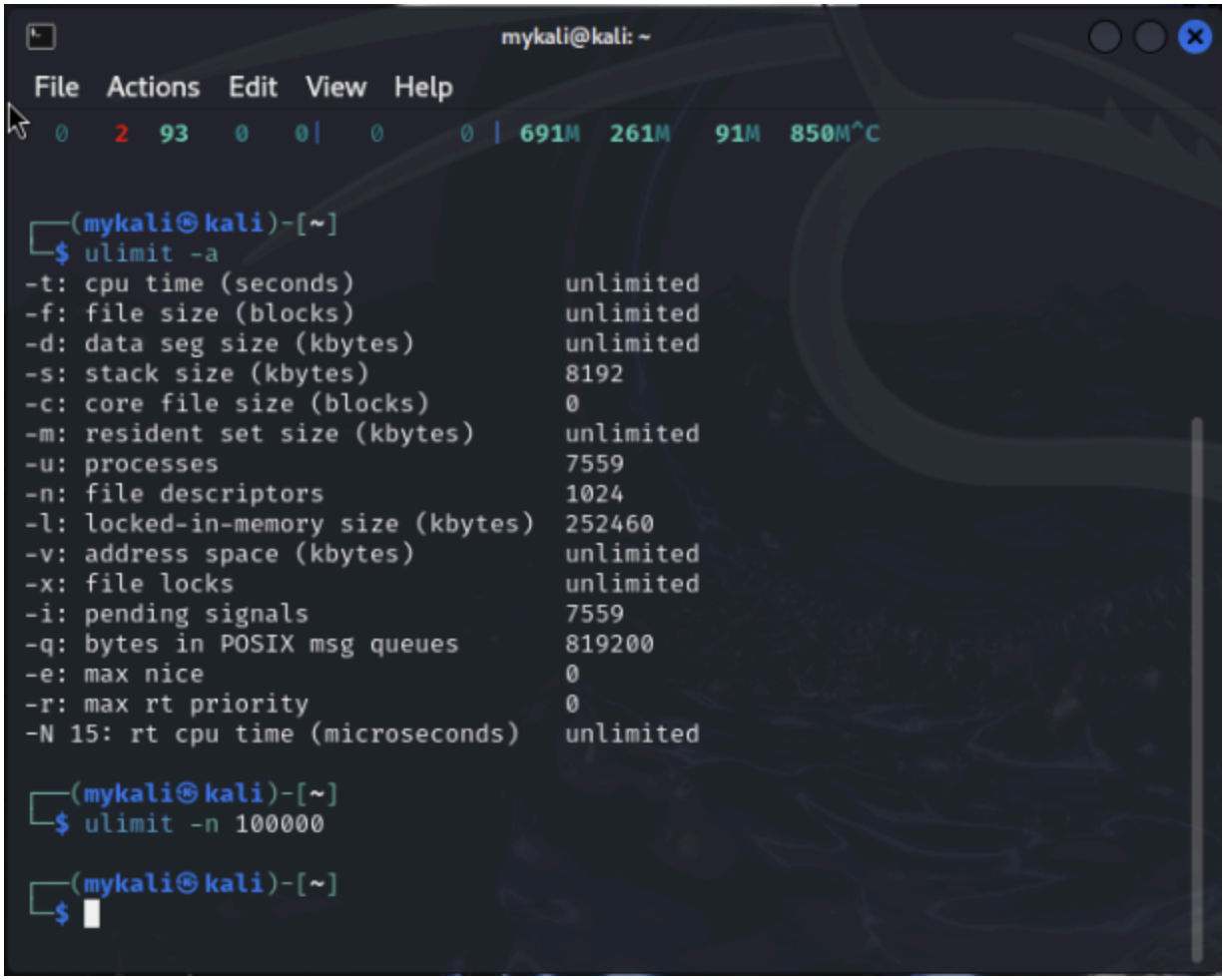
- sudo apt install dstat



Task 4. Check Process Limits:

Use ulimit to check and modify user limits on processes (e.g., maximum number of open files).

- ulimit -a
- ulimit -n 100000 (To modify limit)



Lab 5 : Managing Daemons and Background Services

Task 1. Start and Stop Services:
Use systemctl to start, stop, and restart system services (e.g., systemctl start apache2, systemctl stop nginx).

- sudo systemctl start apache2
- sudo systemctl stop nginx
- sudo systemctl restart apache2

Task 2. Enable/Disable Services on Boot:
Use systemctl enable and systemctl disable to manage whether a service starts on boot.

- sudo systemctl enable apache2
- sudo systemctl disable apache2

Task 3. Check Service Status:
Use systemctl status to check the status of a service (e.g., systemctl status apache2).

- systemctl status apache2

```
mykali@kali: ~
File  Actions  Edit  View  Help

┌──(mykali㉿kali)-[~]
└─$ sudo systemctl start apache2

┌──(mykali㉿kali)-[~]
└─$ sudo systemctl restart apache2

┌──(mykali㉿kali)-[~]
└─$ sudo systemctl enable apache2
Synchronizing state of apache2.service with SysV service script with /usr/lib
/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable apache2

┌──(mykali㉿kali)-[~]
└─$ sudo systemctl disable apache2
Synchronizing state of apache2.service with SysV service script with /usr/lib
/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install disable apache2
Removed "/etc/systemd/system/multi-user.target.wants/apache2.service".

┌──(mykali㉿kali)-[~]
└─$ systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/apache2.service; disabled; pres>
     Active: active (running) since Thu 2025-01-30 00:12:11 IST; 46s ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 69270 (apache2)
```

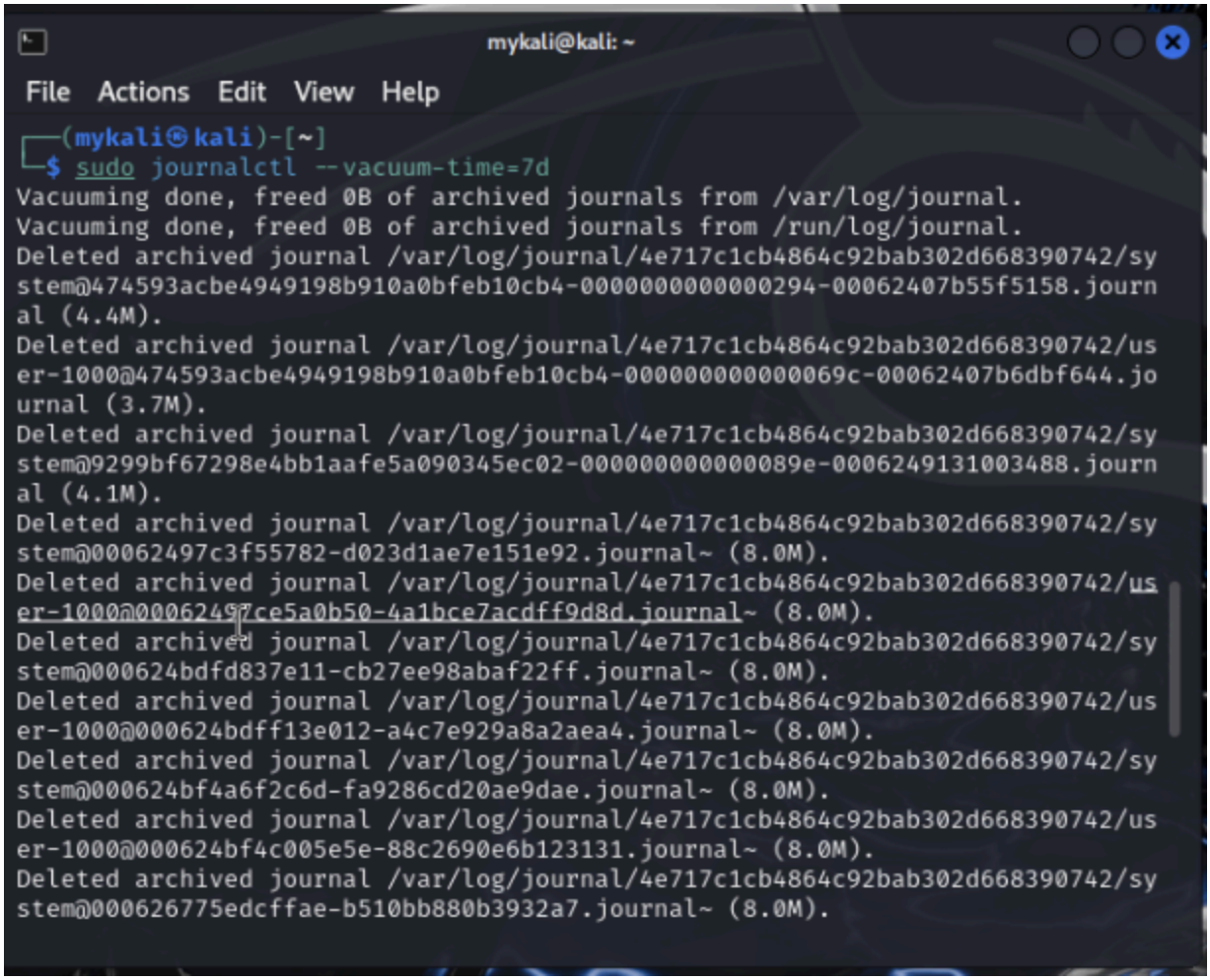
Task 4. Managing Logs for Services:
Use journalctl to check logs for systemd services.
Filter logs for specific services or time periods to troubleshoot issues.

- sudo journalctl -u apache2
- sudo journalctl -u apache2 -f
  NOTE: The flag -f is used to follow real time logs

```
mykali@kali: ~
File  Actions  Edit  View  Help

┌──(mykali㉿kali)-[~]
└─$ sudo journalctl -u apache2
Jan 29 01:25:14 kali systemd[1]: Starting apache2.service - The Apache HTTP >
Jan 29 01:25:14 kali apachectl[33220]: AH00558: apache2: Could not reliably >
Jan 29 01:25:14 kali systemd[1]: Started apache2.service - The Apache HTTP S>
-- Boot 91f325c8b2e543d688f6d0579facb982 --
Jan 29 12:28:41 kali systemd[1]: Starting apache2.service - The Apache HTTP >
Jan 29 12:28:41 kali apachectl[695]: AH00557: apache2: apr_sockaddr_info_get>
Jan 29 12:28:41 kali apachectl[695]: AH00558: apache2: Could not reliably de>
Jan 29 12:28:41 kali systemd[1]: Started apache2.service - The Apache HTTP S>
Jan 29 12:34:14 kali systemd[1]: Stopping apache2.service - The Apache HTTP >
Jan 29 12:34:14 kali apachectl[4365]: AH00558: apache2: Could not reliably d>
Jan 29 12:34:14 kali systemd[1]: apache2.service: Deactivated successfully.
Jan 29 12:34:14 kali systemd[1]: Stopped apache2.service - The Apache HTTP S>
Jan 29 12:34:14 kali systemd[1]: Starting apache2.service - The Apache HTTP >
Jan 29 12:34:14 kali apachectl[4372]: AH00558: apache2: Could not reliably d>
Jan 29 12:34:14 kali systemd[1]: Started apache2.service - The Apache HTTP S>
Jan 29 12:57:22 kali apachectl[16246]: AH00558: apache2: Could not reliably >
Jan 29 12:57:22 kali apachectl[16246]: httpd (no pid file) not running
Jan 29 12:57:22 kali systemd[1]: apache2.service: Deactivated successfully.
Jan 29 12:59:59 kali systemd[1]: Starting apache2.service - The Apache HTTP >
Jan 29 12:59:59 kali apachectl[17636]: AH00558: apache2: Could not reliably >
Jan 29 12:59:59 kali systemd[1]: Started apache2.service - The Apache HTTP S>
Jan 30 00:00:10 kali systemd[1]: Reloading apache2.service - The Apache HTTP>
Jan 30 00:00:10 kali apachectl[61964]: AH00558: apache2: Could not reliably >
Jan 30 00:00:10 kali systemd[1]: Reloaded apache2.service - The Apache HTTP >
```

- sudo journalctl -u apache2 --since today
  NOTE: View logs for today only
- sudo journalctl -u apache2 --since "2024-01-01 10:00:00" --until "2024-01-01 12:00:00"
  NOTE: View logs from a specific date and time
- sudo journalctl -k
  NOTE: View kernel logs
- sudo journalctl --vacuum-time=7d
  NOTE: Clear system logs older than 7 days

```
mykali@kali
File  Actions  Edit  View  Help
┌──(mykali㉿kali)-[~]
└─$ sudo journalctl -u apache2 --since today
Jan 30 00:00:10 kali systemd[1]: Reloading apache2.service - The Apache HTTP>
Jan 30 00:00:10 kali apachectl[61964]: AH00558: apache2: Could not reliably >
Jan 30 00:00:10 kali systemd[1]: Reloaded apache2.service - The Apache HTTP >
Jan 30 00:12:10 kali systemd[1]: Stopping apache2.service - The Apache HTTP >
Jan 30 00:12:10 kali apachectl[69254]: AH00558: apache2: Could not reliably >
Jan 30 00:12:11 kali systemd[1]: apache2.service: Deactivated successfully.
Jan 30 00:12:11 kali systemd[1]: Stopped apache2.service - The Apache HTTP S>
Jan 30 00:12:11 kali systemd[1]: apache2.service: Consumed 3.514s CPU time, >
Jan 30 00:12:11 kali systemd[1]: Starting apache2.service - The Apache HTTP >
Jan 30 00:12:11 kali apachectl[69269]: AH00558: apache2: Could not reliably >
Jan 30 00:12:11 kali systemd[1]: Started apache2.service - The Apache HTTP S>

┌──(mykali㉿kali)-[~]
└─$ sudo journal ctl -k
[sudo] password for mykali:
sudo: journal: command not found

┌──(mykali㉿kali)-[~]
└─$ sudo journalctl -k
Jan 29 12:28:36 kali kernel: Linux version 6.6.15-amd64 (devel@kali.org) (gc>
Jan 29 12:28:36 kali kernel: Command line: BOOT_IMAGE=/boot/vmlinuz-6.6.15-a>
Jan 29 12:28:36 kali kernel: BIOS-provided physical RAM map:
Jan 29 12:28:36 kali kernel: BIOS-e820: [mem 0x0000000000000000-0x0000000000>
Jan 29 12:28:36 kali kernel: BIOS-e820: [mem 0x0000000000100000-0x000000007f>
Jan 29 12:28:36 kali kernel: BIOS-e820: [mem 0x000000007f8ef000-0x000000007f>
```



Lab 6 : Process Scheduling and Prioritization

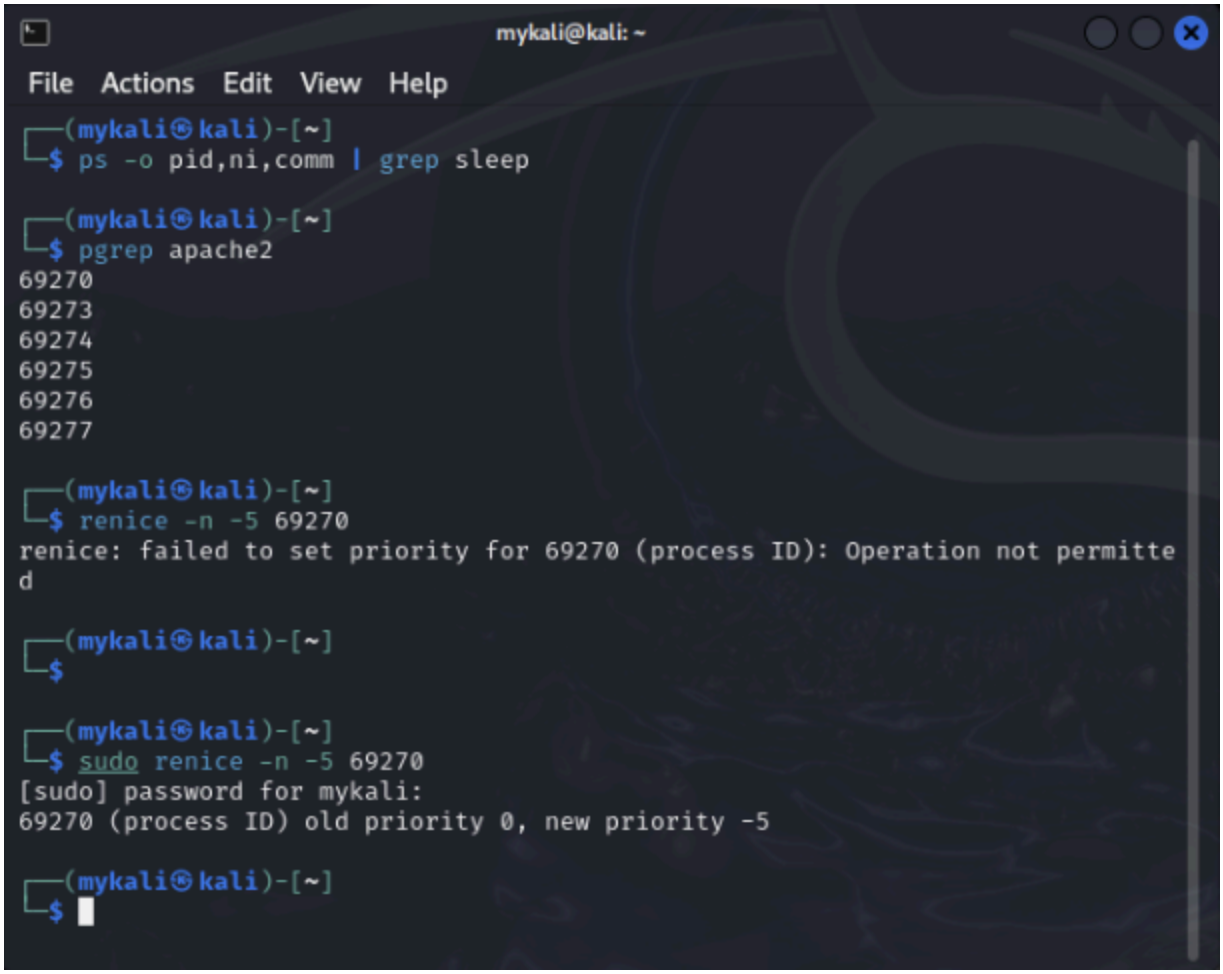
Task 1. Change Process Priority (Nice Value):

Use nice to start a new process with a custom priority level (e.g., nice -n 10 command).

- nice -n 10 sleep 100
- ps -o pid,ni,comm | grep sleep
  NOTE: You can verify the nice value of a process using above command

Use renice to change the priority of an already running process by its PID (e.g., renice -n -5 <PID>).

- renice -n -5 <PID>
- ps -o pid,ni,comm | grep <PID>
  NOTE: Verify the new priority



Task 2. Scheduling Processes:

Use at to schedule a one-time task (e.g., at 09:00 to run a script).

- sudo apt install at
- at 09:00
  NOTE: You will enter an interactive mode, type the commond mentioned below in it (Ctrl+D to exit)
- echo "Hello, World!" > /tmp/hello.txt

Use cron to schedule recurring tasks by adding entries to /etc/crontab or using crontab -e for user-specific jobs.

- crontab -e
- 0 8 * * * [PATH TO SCRIPT]

```
mykali@kali: ~
File  Actions  Edit  View  Help
Processing triggers for kali-menu (2023.4.7) ...

┌──(mykali㉿kali)-[~]
└─$ at 9:00
warning: commands will be executed using /bin/sh
at Fri Jan 31 09:00:00 2025
at> <EOT>
job 1 at Fri Jan 31 09:00:00 2025

┌──(mykali㉿kali)-[~]
└─$ echo "Hello, World!" > /tmp/hello.txt
dquote>

┌──(mykali㉿kali)-[~]
└─$ crontab -e
no crontab for mykali - using an empty one

Select an editor.  To change later, run 'select-editor'.
  1. /bin/nano        <── easiest
  2. /usr/bin/vim.basic
  3. /usr/bin/vim.tiny

Choose 1-3 [1]: 1
No modification made

┌──(mykali㉿kali)-[~]
└─$
```

Task 3. Monitor Process Execution Time:

Use time to measure the execution time of a command or script.

- time ls -l
- time bash [SCRIPT NAME]

```
mykali@kali: ~
File  Actions  Edit  View  Help

┌──(mykali㉿kali)-[~]
└─$ time ls il
ls: cannot access 'il': No such file or directory

real    0.00s
user    0.00s
sys     0.00s
cpu     62%

┌──(mykali㉿kali)-[~]
└─$
```

Lab 7 : Investigating and Debugging Stuck Processes

Task 1. Check for Stuck Processes:
Use ps or top to identify processes that are stuck in a specific state, like D (uninterruptible sleep).

- ps aux | awk '$8 ~ /^D/ { print $0 }'
  NOTE: This was using ps command
- top (Using TOP command)

Task 2. Trace Process Execution:
Use strace to trace the system calls made by a process (e.g., strace -p <PID>).

- strace -p <PID>

Task 3. Analyze Process Core Dumps:
Set up core dumps for processes by configuring /etc/security/limits.conf.

- Edit `/etc/security/limits.conf` and add:

  * soft core unlimited
  * hard core unlimited

- ulimit -c unlimited (To check if the changes are made)
- sudo sysctl -w kernel.core_pattern=/tmp/core.%e.%p
  NOTE: By the above command we set the core dump file pattern

Use gdb to analyze the core dump of a crashed process.

- gdb <executable> <core_file>
- bt (To investigate the cause of crash)

Task 4. Terminate or Kill a Stuck Process:
Use kill -9 to forcefully terminate a stuck process.

- ps aux | grep "<process_name>" (Identify the stuck process)
- kill -9 <PID>

Investigate logs (e.g., /var/log/syslog) for additional clues.

- sudo tail -f /var/log/syslog

Lab 8 : Containerized Processes with Docker

Task 1. Start a Docker Container:
Use docker run to start a container from an image (e.g., docker run -d nginx).

- docker run -d --name my_nginx -p 8080:80 nginx

Task 2. Monitor Processes Inside Containers:
Use docker exec to run commands like top or ps inside a running container to view its processes.

- docker exec -it my_nginx top
- docker exec my_nginx ps aux

Task 3. Stop and Restart Containers:
Use docker stop and docker restart to manage containerized processes.

- docker stop my_nginx
- docker start my_nginx
- docker restart my_nginx

Task 4. Debugging a Stuck Container:
Use docker logs to view logs and diagnose issues in a container.
Check container resource usage using docker stats.

- docker logs -f my_nginx
- docker stats my_nginx
- docker inspect my_nginx

Lab 9 : Process Resource Usage and Optimization

Task 1. Analyze Resource Usage:
Use ps aux --sort=-%mem or top to find the processes consuming the most memory and CPU.

- Using the ps aux:
- ps aux --sort=-%mem | head -n 6
  NOTE: Above command find top 5 memory consuming processes
- ps aux --sort=-%cpu | head -n 6
  NOTE: Above command find top 5 memory consuming processes
- Using top:
- top

Task 2. Optimize Memory Usage:
Identify memory leaks or inefficient memory usage with valgrind or smem.

  # Check process memory with smem
- smem -s rss -r | head -10

  # Analyze memory leaks with valgrind (for C/C++ programs)
- valgrind --leak-check=full ./your_application

  # Docker-specific memory checks
- docker stats --no-stream | sort -k4 -h -r # Sort by memory usage

Task 3. Optimize CPU Usage:
Use cpulimit or nice to adjust CPU resource allocation for specific processes.

```
# Limit a process to 50% CPU usage
- cpulimit -l 50 -p <PID>

# Start a process with low priority
- nice -n 19 ./cpu_intensive_script.sh

# Real-time CPU monitoring
- mpstat -P ALL 1  # Show per-core usage every second
```

Task 4. Tune System Parameters:
Tune kernel parameters related to process management using sysctl (e.g., sysctl -w vm.swappiness=10).

```
# Reduce swap usage tendency
- sudo sysctl -w vm.swappiness=10

# Increase open files limit
- sudo sysctl -w fs.file-max=100000

# Better TCP performance
- sudo sysctl -w net.core.somaxconn=65535
- sudo sysctl -w net.ipv4.tcp_fin_timeout=15

# Make changes permanent
- sudo nano /etc/sysctl.conf  # Add parameters here
- sudo sysctl -p
```