Basic Monitor

This challenge is divided in two tasks, the first one having you research how to monitor a Linux system as well as what to look for when doing so. You will have to take note of all your findings in a text file (EX: *markdown*) while being as exhaustive as possible (*what to monitor*, *how to monitor it*, *commands used*, *...*). Try to answer, but don't limit yourself to, the questions below to guide you through the research process:

1. What are the main areas of concern when monitoring a system? (EX: *CPU load*, *disk usage*, ...)
   CPU Load: Monitor CPU usage to ensure the system isn't overloaded. High load averages can indicate performance bottlenecks.
   Memory Usage: Keeping an eye on RAM usage helps in identifying memory leaks or processes that consume excessive memory.
   Disk Usage: Monitor disk space and I/O operations to avoid running out of space, which can lead to system failures.
   Network Traffic: Check network interfaces to ensure no unusual spikes or drops in traffic, which could indicate problems or security breaches.
   Processes: Monitor running processes to identify any that are consuming too many resources or behaving unexpectedly.
   System Logs: Regularly check logs to detect errors, warnings, or security issues.

2. How can you check what are the most memory intensive running processes?

   Use commands like `top`, `htop`, or `ps aux --sort=-%mem` to view the most memory-consuming processes.
3. What are log files? Where can you find them on a typical Linux system?

   What Are They?: Log files record system events, errors, and other significant activities.

   Where to Find Them: Typically located in `/var/log/`. Important logs include:
   - `syslog` or `messages`: General system logs.
   - `auth.log` or `secure`: Authentication logs.
   - `dmesg`: Kernel ring buffer logs, often hardware-related.
   - `journalctl`: For systems using `systemd`, to query logs.

4. How can you check who were the last connected users, what they did, when they left?
   - Use commands like `last`, `w`, and `who` to view user login history and active sessions.
   - For command history, check `.bash_history` in user home directories (note: this might be considered invasive and should be done with caution).

5. What are the different metrics of health and performance of a system?

   CPU Load: `uptime`, `top`, `htop`, `mpstat`.

   Memory Usage: `free -h`, `vmstat`, `top`, `htop`.

   Disk Usage: `df -h`, `du -sh`, `iostat`.

   Network Traffic: `ifconfig`, `netstat`, `iftop`, `nload`, `vnstat`.

   Processes: `ps aux`, `top`, `htop`.

6. How can you check the uptime of a machine?

   Use the `uptime` command to see how long the system has been running.

7. How can you assess the network traffic?

   Commands like `iftop`, `nload`, `vnstat`, `ip -s link`, and `ss` help monitor network traffic and connections.

The second task is meant to serve as practice and will have you, in a different file, write a report with as many relevant information (*what would make sense in a report*) as you can muster on a system you manage. It most preferably would be a remote machine, but it can also be your local machine as this is just practice.

The Report

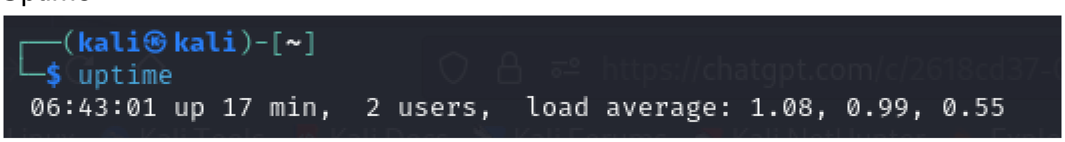
Research Notes and System Report

Linux System Monitoring Research

- Main Areas of Concern

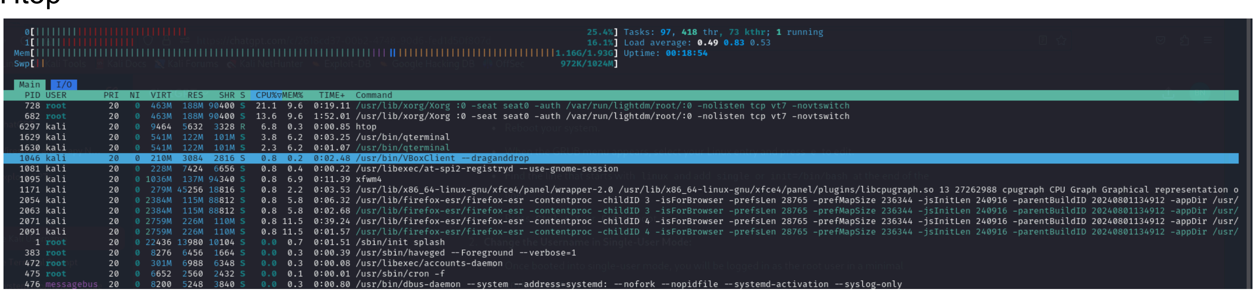
  \- CPU Load:

  Uptime



The system's been up for 17 minutes with a load average of 1.08, 0.99, and 0.55 over the past 1, 5, and 15 minutes, respectively. It looks like the system is handling its load well right now. If you're monitoring performance or running some tests, these numbers seem to be in a good range for your setup.

Htop



Top Section: System Summary

1. **Bars (Load, CPU, Memory, and Swap Usage)**
   - **Load Average**: The bars at the very top show load averages for 1, 5, and 15 minutes. The load average indicates the number of processes that are either running or waiting for CPU time.
   - **CPU Usage (25.4%)**: The colored bar represents CPU usage. The percentage shows how much of the CPU's capacity is being used.
     - **Blue/Green**: Lower-priority (nice) processes.
     - **Red**: Kernel/system processes.
     - **Orange/Yellow**: Normal priority processes.
   - **Memory Usage (16.1%)**: Shows the percentage of RAM being used.
   - **Swap Usage**: Shows the usage of swap space. Swap is disk space used when RAM is full.
2. **System Information**
   - **Tasks**: Number of running, sleeping, stopped, and zombie processes.
   - **Load Average**: The average system load for the last 1, 5, and 15 minutes.
   - **Uptime**: How long the system has been running since the last boot.
   - **Memory and Swap Usage**: Real-time memory (RAM) and swap usage, with numerical and graphical representations.

Bottom Section: Process List

This section lists all running processes, with each row representing one process. Here's how to read it:

1. **PID (Process ID)**: The unique identifier for each running process.
2. **USER**: The user who owns the process (e.g., `root` or `kali`).
3. **PRI (Priority)**: The priority of the process. Lower values mean higher priority.
4. **NI (Nice Value)**: The nice value, which affects process scheduling priority. The lower the nice value, the higher the priority.
5. **VIRT (Virtual Memory)**: The total virtual memory used by the process.
6. **RES (Resident Memory)**: The non-swapped physical memory used by the process.
7. **SHR (Shared Memory)**: The amount of shared memory used by the process.
8. **S (State)**: The current state of the process:
   - `R`: Running
   - `S`: Sleeping
   - `D`: Uninterruptible sleep
   - `Z`: Zombie
   - `T`: Stopped
9. **CPU%**: The percentage of CPU time used by the process.
10. **MEM%**: The percentage of physical RAM used by the process.
11. **TIME+**: The total CPU time the process has used since it started.
12. **Command**: The command that started the process.

Highlighted Rows

- The highlighted row indicates the currently selected process (`VBoxClient --draganddrop`). This process is using very little CPU (0.2%) and a small amount of memory (3084 KB of resident memory).

Mpstat

```
┌──(kali㉿kali)-[~]
└─$ mpstat
Linux 6.8.11-amd64 (kali)       09/04/2024      _x86_64_        (2 CPU)

06:45:28 AM  CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest  %gnice   %idle
06:45:28 AM  all    5.16    0.00    6.18    0.38    0.00    0.68    0.00    0.00    0.00   87.61
```

From your `mpstat` output:

- **%usr** (User CPU time): 5.16% – This indicates that the CPU is spending 5.16% of its time on user processes.
- **%nice** (Nice CPU time): 0.00% – No CPU time is being spent on processes with adjusted priorities (nice value).
- **%sys** (System CPU time): 6.18% – The CPU is spending 6.18% of its time on kernel processes.
- **%iowait** (I/O wait): 0.38% – The CPU is waiting 0.38% of the time for I/O operations to complete.
- **%irq** (Hardware interrupt): 0.00% – No CPU time is spent handling hardware interrupts.
- **%soft** (Software interrupt): 0.68% – The CPU is spending 0.68% of its time handling software interrupts.
- **%steal** (Steal time): 0.00% – No time is being "stolen" by the hypervisor in a virtualized environment.
- **%guest** (Guest time): 0.00% – The CPU is not running virtual machines.
- **%gnice** (Guest nice time): 0.00% – No time is spent on nice-adjusted virtual machines.
- **%idle** (Idle time): 87.61% – The CPU is idle 87.61% of the time, indicating low usage.

Overall, your CPU is under light load with significant idle time, suggesting that your system isn't very busy at the moment.

2. Checking Memory-Intensive Processes
- **Commands**: `ps aux --sort=-%mem`, `top`, `htop`

```
┌──(kali㉿kali)-[~]
└─$ ps aux --sort=-%mem
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
kali        1734  5.0 22.5 11551728 455352 ?     Sl   06:35   1:18 /usr/lib/firefox-esr/firefox-esr
kali        2071  3.5 10.4 2799360 211832 ?      Sl   06:36   0:55 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 4 -isForBrowser -prefsLen 28765 -prefMapSize 236344 -jsInitLen 240916 -parentBuildID 20240801134912 -appDir /usr/
root         682  7.4  9.1 467616 185748 tty7    Ssl+ 06:25   2:43 /usr/lib/xorg/Xorg :0 -seat seat0 -auth /var/run/lightdm/root/:0 -nolisten tcp vt7 -novtswitch
kali        1095  1.4  6.9 1061184 140504 ?      Rl   06:35   0:23 xfwm4
kali        1910  0.1  6.2 2446160 127096 ?      Sl   06:35   0:02 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 1 -isForBrowser -prefsLen 26367 -prefMapSize 236344 -jsInitLen 240916 -parentBuildID 20240801134912 -appDir /usr/
kali        1629  0.3  6.1 554472 125024 ?       Sl   06:35   0:05 /usr/bin/qterminal
kali        2054  2.0  5.9 2443612 120928 ?      Sl   06:36   0:32 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 3 -isForBrowser -prefsLen 28765 -prefMapSize 236344 -jsInitLen 240916 -parentBuildID 20240801134912 -appDir /usr/
kali        2086  0.0  5.6 2473148 115232 ?      Sl   06:36   0:01 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 5 -isForBrowser -prefsLen 28765 -prefMapSize 236344 -jsInitLen 240916 -parentBuildID 20240801134912 -appDir /usr/
kali        1973  0.0  4.8 2434348 98284 ?       Sl   06:36   0:00 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 2 -isForBrowser -prefsLen 31531 -prefMapSize 236344 -jsInitLen 240916 -parentBuildID 20240801134912 -appDir /usr/
kali        2192  0.0  3.5 2402448 71104 ?       Sl   06:36   0:00 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 7 -isForBrowser -prefsLen 28765 -prefMapSize 236344 -jsInitLen 240916 -parentBuildID 20240801134912 -appDir /usr/
kali        2229  0.0  3.5 2402452 71056 ?       Sl   06:36   0:00 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 8 -isForBrowser -prefsLen 28765 -prefMapSize 236344 -jsInitLen 240916 -parentBuildID 20240801134912 -appDir /usr/
kali        2174  0.0  3.5 2402448 71020 ?       Sl   06:36   0:00 /usr/lib/firefox-esr/firefox-esr -contentproc -childID 6 -isForBrowser -prefsLen 28765 -prefMapSize 236344 -jsInitLen 240916 -parentBuildID 20240801134912 -appDir /usr/
kali        1164  0.1  3.2 482352 65568 ?        Sl   06:35   0:01 xfdesktop
kali        1232  0.0  2.5 589308 51848 ?        Sl   06:35   0:00 /usr/bin/python3 /usr/bin/blueman-applet
kali        1171  0.4  2.2 285912 45256 ?        Sl   06:35   0:07 /usr/lib/x86_64-linux-gnu/xfce4/panel/wrapper-2.0 /usr/lib/x86_64-linux-gnu/xfce4/panel/plugins/libcpugraph.so 13 27262988 cpugraph CPU Graph Graphical representation o
kali        1249  0.6  2.2 618940 45132 ?        Sl   06:35   0:10 nm-applet
```

From your `ps aux` output sorted by memory usage, it's clear that Firefox is using a significant amount of RAM on your system. Here's a brief overview of the processes consuming the most memory:

- **Firefox ESR Main Process**:
  - **PID 1734**: Using 22.5% of memory.
- **Firefox ESR Content Processes**:
  - **PID 2071**: 10.4% of memory.
  - **PID 2054**: 5.9% of memory.
  - **PID 1910**: 6.2% of memory.
  - **PID 2086**: 5.6% of memory.

        ➢ **PID 1973**: 4.8% of memory.
        ➢ **PID 2192**: 3.5% of memory.
        ➢ **PID 2229**: 3.5% of memory.
        ➢ **PID 2174**: 3.5% of memory.

❖ **Xorg (X Window System)**:
        ➢ **PID 682**: Using 9.1% of memory.

❖ **xfwm4 (Xfce Window Manager)**:
        ➢ **PID 1095**: Using 6.9% of memory.

❖ **qterminal**:
        ➢ **PID 1629**: Using 6.1% of memory.

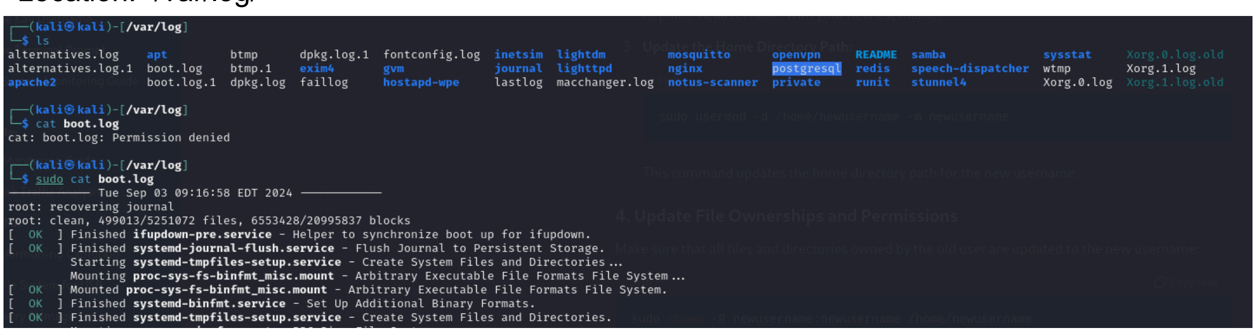
❖ **xfdesktop (Xfce Desktop)**:
        ➢ **PID 1164**: Using 3.2% of memory.

❖ **Blueman Applet**:
        ➢ **PID 1232**: Using 2.5% of memory.

Insights:

- **Firefox**: It seems to be the primary consumer of memory on your system, with its various content processes adding up to a substantial amount. Firefox tends to use a lot of memory, especially if you have multiple tabs open or if you are running various extensions.
- **Xorg**: This is the display server, which also uses a significant amount of memory. This is normal but can vary based on graphical settings and workloads.
- **xfwm4 and xfdesktop**: These are components of the Xfce desktop environment. Their memory usage is relatively moderate.
- **qterminal**: Terminal emulators usually have a small memory footprint but can grow with the amount of data or history they manage.
- **Blueman Applet**: A small footprint for a system tray application that manages Bluetooth.

3. Log Files
- Location: `/var/log/`



4. Checking Last Connected Users
- Commands**: `last`, `who`, `w`
Last

```
┌──(kali㉿kali)-[/var/log]
└─$ last
lightdm  tty8         :1               Wed Sep  4 06:56 - 07:02  (00:05)
root     pts/0                         Wed Sep  4 06:39 - 06:42  (00:02)
kali     tty7         :0               Wed Sep  4 06:35 - still logged in
lightdm  tty7         :0               Wed Sep  4 06:25 - 06:35  (00:09)
lightdm  tty8         :1               Wed Sep  4 05:03 - 06:15  (01:11)
lightdm  tty8         :1               Wed Sep  4 04:55 - 04:57  (00:02)
lightdm  tty8         :1               Tue Sep  3 09:38 - 04:46  (19:08)

/var/lib/wtmpdb/wtmp.db begins Tue Sep  3 09:38:52 2024
```

Your `last` command output provides a log of recent login sessions and system activity. Here's a breakdown of the entries:

1. **Current Session**:
   - **kali**: Logged in at `tty7` (likely your primary display) from `Wed Sep 4 06:35` and is still logged in.
2. **Recent Logins**:
   - **lightdm** (Display Manager) logged into `tty8` from `Wed Sep 4 06:56` to `Wed Sep 4 07:02` (5 minutes).
   - **root** logged into `pts/0` (a terminal session) from `Wed Sep 4 06:39` to `Wed Sep 4 06:42` (2 minutes).
   - **lightdm** at `tty7` from `Wed Sep 4 06:25` to `Wed Sep 4 06:35` (9 minutes).
   - **lightdm** at `tty8` from `Wed Sep 4 05:03` to `Wed Sep 4 06:15` (1 hour 11 minutes).
   - **lightdm** at `tty8` from `Wed Sep 4 04:55` to `Wed Sep 4 04:57` (2 minutes).
   - **lightdm** at `tty8` from `Tue Sep 3 09:38` to `Wed Sep 4 04:46` (19 hours 8 minutes).

W

```
┌──(kali㉿kali)-[/var/log]
└─$ w
 07:18:00 up 52 min,  2 users,  load average: 0.27, 0.33, 0.40
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
kali              -                06:35     ?     0.00s  0.21s /usr/lib/systemd/systemd --user
kali              -                06:35     ?     0.00s  0.07s lightdm --session-child 13 24
```

The `w` command output provides information about who is currently logged into the system and what they are doing. Here's a breakdown of the information:

Current System Status:

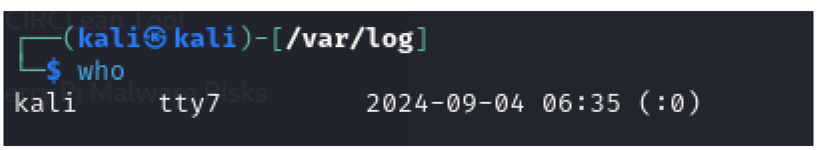
- **Time**: `07:18:00`
- **Uptime**: 52 minutes
- **Users Logged In**: 2
- **Load Average**:
  - 1 minute: 0.27
  - 5 minutes: 0.33
  - 15 minutes: 0.40

The load averages are well within acceptable ranges, indicating that the system is not under heavy load.

Logged-In Users:

1. **kali**:
    - **TTY**: `-` (indicates that this session is not tied to a terminal; often used for GUI or system services)
    - **FROM**: `-` (since this is a graphical session or system service, there's no remote origin)
    - **LOGIN@**: `06:35` (the time when the user session started)
    - **IDLE**: `?` (not applicable in this context)
    - **JCPU**: `0.00s` (CPU time used by all processes attached to the terminal)
    - **PCPU**: `0.21s` (CPU time used by the current process)
    - **Command**: `/usr/lib/systemd/systemd --user`
        - This indicates a user service managed by `systemd` is running under your user session.
2. **kali** (again):
    - **TTY**: `-`
    - **FROM**: `-`
    - **LOGIN@**: `06:35`
    - **IDLE**: `?`
    - **JCPU**: `0.00s`
    - **PCPU**: `0.07s`
    - **Command**: `lightdm --session-child 13 24`
        - This shows the LightDM session process running, indicating that it's managing the current graphical session.

Who



The `who` command output shows the currently logged-in users and their terminal details:

Current User Session:

- **Username**: `kali`
- **Terminal**: `tty7`
- **Login Time**: `2024-09-04 06:35`
- **Display**: `:0` (indicating the graphical session is on display `:0`)

5. System Health and Performance Metrics
- CPU: Commands
- **CPU Load**: `uptime`, `top`, `htop`, `mpstat`.
Memory Usage: `free -h`, `vmstat`, `top`, `htop`.

Disk Usage: `df -h`, `du -sh`, `iostat`.
Network Traffic: `ifconfig`, `netstat`, `iftop`, `nload`, `vnstat`.
Processes: `ps aux`, `top`, `htop`.

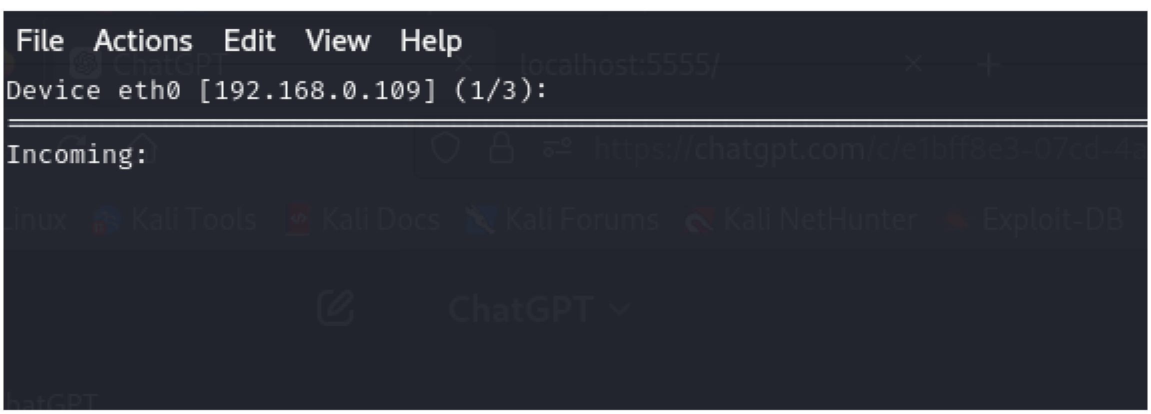
6. Checking Uptime
- Command: `uptime`
Already done above

7. Assessing Network Traffic
- Commands: `iftop`, `nload`, `vnstat`
Iftop

Here are some key features of `iftop`:

- **Real-Time Traffic Monitoring**: `iftop` shows the bandwidth usage of different network connections in real time.
- **Connection Details**: It lists connections along with their source and destination IP addresses, ports, and the amount of data being transmitted and received.
- **Traffic Summaries**: You can see the total incoming and outgoing traffic and can sort the connections based on various criteria like data transfer rate.
- **Interactive Interface**: The tool provides an interactive, curses-based interface where you can filter and customize the view according to your needs.

Nload



`nload` is another useful network monitoring tool, but it has some distinct features compared to `iftop`. While `iftop` focuses on displaying detailed information about individual network connections and their bandwidth usage, `nload` provides a more general overview of network traffic on a per-interface basis.

Here's what you need to know about `nload`:

Key Features:

- **Traffic Visualization**: `nload` displays incoming and outgoing traffic separately in graphical format. It uses simple charts to show the data rates and traffic volume over time.
- **Per-Interface Monitoring**: It monitors and displays network traffic statistics for individual network interfaces. This allows you to see how much traffic is passing through each network interface on your system.
- **Real-Time Data**: Like `iftop`, `nload` provides real-time monitoring of network traffic. However, instead of detailed connection information, it provides a high-level overview.
- **Historical Data**: `nload` can show historical traffic data as well, allowing you to observe traffic trends over time.
- **User Interface**: It features a user-friendly, text-based interface that is easy to understand and navigate. The display includes graphs that represent traffic volumes over time.

Vnstat

```
┌──(kali㉿kali)-[~]
└─$ vnstat

                     rx      /      tx      /     total    /   estimated
eth0: No data. Timestamp of last update is same 2024-09-04 07:44:18 as of database creation.
eth1: No data. Timestamp of last update is same 2024-09-04 07:44:18 as of database creation.
```

The message from `vnstat` indicates that it hasn't collected any network data since the database was created or last updated. This situation usually happens when `vnstat` hasn't had a chance to monitor the network interfaces properly or when the service hasn't been running long enough to gather data.

Ss

```
┌──(kali㉿kali)-[~]
└─$ ss
Netid    State     Recv-Q    Send-Q                  Local Address:Port             Peer Address:Port
u_str    ESTAB     0         0                                 * 11738                         * 0
u_str    ESTAB     0         0         /run/dbus/system_bus_socket 9062                        * 10049
u_str    ESTAB     0         0                                 * 8613                          * 8614
u_str    ESTAB     0         0                                 * 11709                         * 10848
u_str    ESTAB     0         0         /run/systemd/journal/stdout 9858                        * 9857
u_str    ESTAB     0         0                                 * 10956                         * 10957
u_str    ESTAB     0         0                                 * 10408                         * 10409
u_str    ESTAB     0         0         /run/systemd/journal/stdout 10277                       * 10275
u_str    ESTAB     0         1536      /run/dbus/system_bus_socket 9411                        * 9410
u_str    ESTAB     0         0                                 * 11722                         * 10860
u_str    ESTAB     0         0         /run/user/1000/at-spi/bus_0 11707                       * 11706
```

Unix domain sockets are used for inter-process communication (IPC) on the same host and can be seen in various states. Here's a quick breakdown of the information shown:

Netid: Indicates the type of socket. In this case, `u_str` denotes Unix domain stream sockets.

State: The state of the socket connection. `ESTAB` means "established," which indicates that the connection is open and ready for communication.

Recv-Q and **Send-Q**: The receive and send queue sizes. A value of `0` means there is no pending data in either queue.

Local Address

: The local address and port the socket is bound to. For Unix domain sockets, this is typically a file path.

Peer Address
: The address and port of the peer socket. For Unix domain sockets, this is often another file path or `*` if the peer is not specified.

- Here's what each entry represents:
- `* 11738` - A Unix domain socket in the `ESTAB` state with no specific local address or peer address.
- `/run/dbus/system_bus_socket 9062` - The DBus system bus socket.
- `* 8613` - Another Unix domain socket in the `ESTAB` state with no specific local address or peer address.
- `/run/systemd/journal/stdout 9858` - A socket related to systemd journal output.
- `* 10956` - Another Unix domain socket in the `ESTAB` state with no specific local address or peer address.
- `/run/systemd/journal/stdout 10277` - Another socket related to systemd journal output.
- `/run/dbus/system_bus_socket 9411` - Another instance of the DBus system bus socket.