RPS
learn..evolve

WIPRO NGA Program – LSP Batch

Capstone Project Presentation – 31 July 2024

Project Title Here - Linux device driver to print system metrics

Presented by - Tatiparthi Aruna

Objective

- The main objective is to create a reliable tool for system diagnostics and performance optimization.
- A Linux device driver that retrieves and prints critical system metrics such as CPU usage, memory utilization, disk I/O, and network activity.
- This driver should provide real-time monitoring, enabling users and system administrators to assess the performance and health of the system.
- The driver will interact with the kernel to gather data and output it in a user-friendly format, ensuring minimal overhead and maximum accuracy..

Motivation

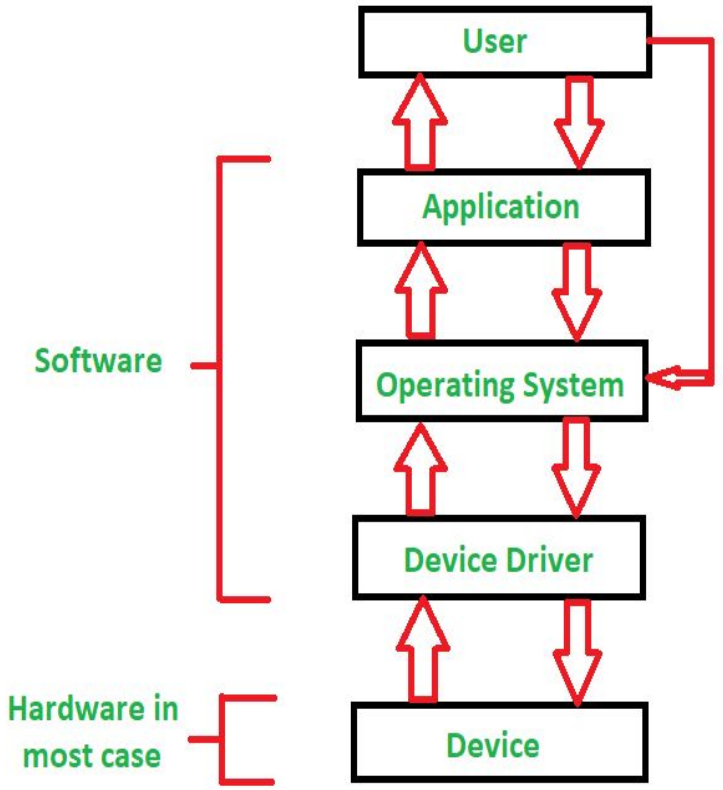
- Major motivation is to enhance system security by enabling real-time monitoring of unusual or suspicious activity.

- By tracking metrics such as
  - Unexpected spikes in CPU usage
  - Unusual memory consumption
  - Abnormal network traffic

- The device driver can help identify potential security threats or system intrusions, allowing for quicker detection and response.

Linux device drivers

- Linux device drivers are specialized software components that allow the Linux kernel to communicate with hardware devices.
- They act as a bridge between the operating system and hardware, providing the necessary interfaces and abstractions for the system to utilize hardware functionalities, such as storage devices, network interfaces.



Types of device drivers

- Character devices – These devices transmit the data character by characters, like a mouse or a keyboard.
- Block devices – These devices transfer unit of data storage called a block, USB drives, hard drives, and CD ROMs.

Applications

- Performance Monitoring
- Troubleshooting and Diagnostics
- Security Monitoring
- Capacity Planning
- Real-Time Alerts

What is system metrics

System metrics are quantitative data points that provide insights into the performance and status of a computer system. They help in monitoring, diagnosing, and optimizing the system's operations. Common system metrics include:

- **CPU Usage**: Measures the percentage of CPU capacity being used by processes. It helps assess how much of the processor's power is being utilized and whether there are performance bottlenecks.
- **Memory Utilization**: Indicates the amount of physical and virtual memory being used. This metric helps in understanding memory allocation, detecting memory leaks, and ensuring there is enough memory available for applications.
- **Network Activity**: Measures data transmission over the network interfaces, including metrics like bandwidth usage, packet loss, and network latency. It is essential for assessing network performance and detecting potential issues.

To print system metrics using a character device driver in Linux, you can follow these steps:

1.Create a character device driver

- **Define the Driver Structure**: Create a struct file_operations to specify the operations your driver will support, such as open, read, write, and release.
- **Register the Device**: Use register_chrdev or alloc_chrdev_region to register the character device and obtain a major number. Implement a device_create function to create a device file in /dev.

2. Implement File Operations

- **Open**: Implement the open function to initialize or prepare the device for use.
- **Read**: Implement the read function to gather system metrics. This function will:
  - Access system metrics like CPU usage, memory utilization, disk I/O, etc., by interfacing with kernel APIs, such as procfs or sysfs.
  - Format the metrics into a buffer that can be read by user-space applications.
  - Copy the data from the kernel space to user space using functions like copy_to_user.

3. Gather System Metrics

- **CPU Usage**: Read from /proc/stat or use kernel functions to get CPU statistics.
- **Memory Utilization**: Access /proc/meminfo for memory statistics.
- **Disk I/O**: Use the /proc/diskstats file or the blkdev_get_stats function for disk I/O information.
- **Network Activity**: Obtain network metrics from /proc/net/dev.

4. Communicate with User Space

- User-space applications can open the device file (e.g., /dev/sys_metrics) and use standard file operations like read to obtain the metrics.
- Implement the ioctl operation if you need to provide additional control or configuration options.

5. Handle Cleanup

- Implement the release function to clean up resources when the device is no longer in use.
- Use unregister_chrdev or unregister_chrdev_region during the module exit routine to unregister the device.

Thank you