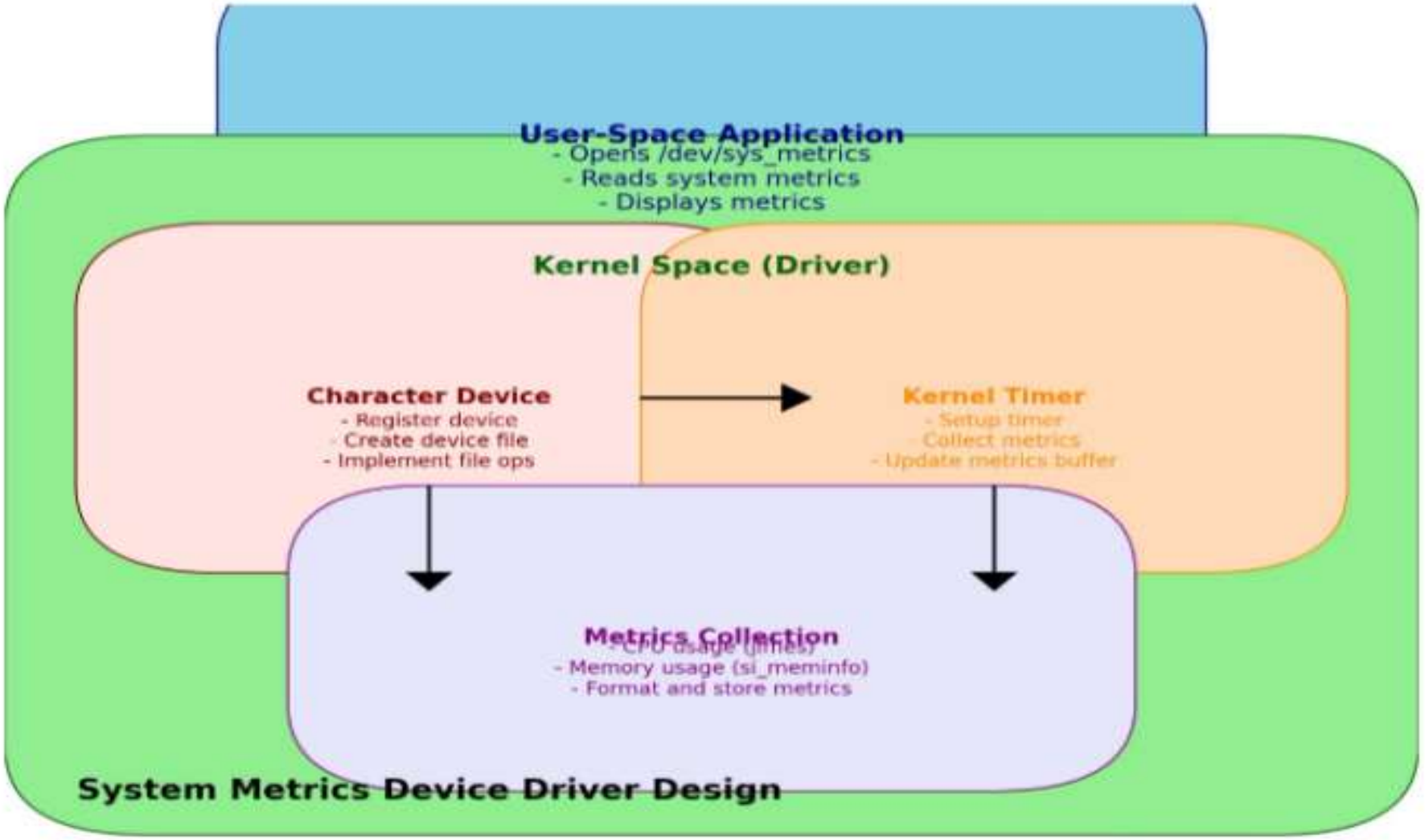
WIPRO NGA Program – LDD Batch

Capstone Project Presentation – 31 July 2024

Project Title Here - Linux Character Device Driver for System Metrics

Presented by – Sourav Dey

SYSTEM DESIGN



PROJECT OVERVIEW

This project aims to provide hands-on experience in developing a Linux device driver to print system metrics. In this project we will learn how to interact with the Linux kernel, read system metrics, and output these metrics via a character device interface.

INTRODUCTION

This project involves developing a Linux character device driver to retrieve and output system metrics such as CPU usage, memory usage, and disk I/O metrics. The device driver can be loaded and unloaded from the kernel and provides a character device interface for user interaction.

MOTIVATION

The motivation behind this project is to develop a Linux character device driver that provides real-time monitoring of system performance metrics. This tool helps in proactive resource management by offering insights into CPU, memory, and disk usage. Additionally, it serves as a valuable learning experience in Linux kernel programming and practical system monitoring.

PROJECT SCOPE

- Create a Linux character device driver to print system metrics.
- Retrieve various system metrics such as CPU usage, memory usage, and disk I/O.
- Output the retrieved metrics through a character device.

Various Application Tools That Are Used In This Project:-

1. Kernel Development Tools:
   - GCC(GNU Compiler Collection)
   - Make
   - Insmod and rmmod
   - dmesg
2. System Monitoring Tools:
   - /proc Filesystem
   - Cat Command
   - IOCTL(Input/Output Control)
3. Memory Management Tools:
   - Kmalloc and kfree
   - timer_list and mod_timer

4. Debugging and Testing Tools:
    - Valgrind
    - Shell Scripts

By integrating these various tools and applications, the project ensures comprehensive development, testing, and debugging of the Linux character device driver for system metrics. This combination facilitates efficient system monitoring and provides a reliable interface for accessing crucial system performance data.

The Modules That I Have Worked On This Project:-

1. Device Driver Module:
    - Character Device Interface
    - File Operations
2. System Metrics Collection Module:
    - System Metrics Retrieval
    - Data Formatting
3. Timer Functionality Module:
    - Kernel Timer
    - Timer Callback
4. Memory Management Module:
    - Dynamic Memory Allocation
    - Resource Cleanup

5. Error Handling Module:
    - Operation Validations
    - Kernel Logging
6. User Interface Module:
    - Command Interface
    - User Manual

By working on these modules, the project ensures a comprehensive and functional Linux character device driver that outputs system metrics, offering valuable insights into system performance and resource usage.

The List Of Functions That Used In This Project:-

Device Driver Functions:-

1. Module Initialization and Cleanup:

- __init sys_metrics_init(void)

- Initializes the device driver, registers the character device, sets up the device class, creates the device file, and starts the timer.

- __exit sys_metrics_exit(void)

- Cleans up resources, unregisters the character device, destroys the device file and class, and stops the timer.

File Operations Functions:-

1. Open and Release Operations:

- int dev_open(struct inode *inodep, struct file *filep)

- Handles the opening of the device file.

- int dev_release(struct inode *inodep, struct file *filep)

- Handles the closing of the device file.

2. Read and Write Operations:

- ssize_t dev_read(struct file *filep, char *buffer, size_t len, loff_t *offset)

- Handles reading from the device file, transferring system metrics data to user space.

- ssize_t dev_write(struct file *filep, const char *buffer, size_t len, loff_t *offset)

- Handles writing to the device file, which is not supported in this project and returns an error.

Timer Functions:-

1. Timer Callback:
   - ➢ void metrics_timer_callback(struct timer_list *timer)

   - Callback function that is executed when the timer expires, triggering the retrieval of system metrics and resetting the timer.

By using these functions, the project achieves its goal of developing a Linux character device driver that outputs system metrics, providing a useful tool for system monitoring and performance analysis.

Future Amendments Addressing Common Challenges:-

1. Handling Increased Load:

- **Challenge**: As systems become more complex, the device driver might need to handle larger volumes of data or more frequent metrics collection.
- **Amendment**: Optimize the driver to efficiently handle high-frequency data collection and large datasets, possibly incorporating asynchronous processing or batch data handling.

2. Extended Metrics:

- **Challenge**: Users might require additional or more granular metrics beyond those initially implemented.
- **Amendment**: Extend the metrics collection capabilities to include more detailed or diverse system data, such as network statistics or application-specific metrics.