**Custom Kernel Module Development**

**Team Members:**

|  |  |  |
| --- | --- | --- |
| S.No. | Name | USN |
| 1. | Saksham Gupta | 1RV23CY047 |
| 2. | Sarthak Lakhotia | 1RV23CY049 |

**Objective:**

The project aims to design and implement a custom kernel module for the Linux operating system. This module will interact with the kernel to provide functionality for managing a virtual device, demonstrating an understanding of kernel architecture, memory management, and device drivers.

**Problem Definition:**

Operating systems rely on kernel modules for extending functionality without requiring kernel recompilation. Developing a custom kernel module provides practical insights into OS internals, such as device management, memory allocation, and system call handling. This project focuses on implementing a kernel module to manage a virtual character device, allowing users to interact with the device through basic read/write operations.

**Proposed Features:**

1. **Virtual Character Device:**

* Create a virtual character device using register\_chrdev().
* Enable user interaction with the device using standard I/O operations (read, write).

1. **Device Driver Operations:** Implement open, close, read, and write operations for the device.
2. **Buffer Management:** Provide a buffer for data storage with appropriate memory management techniques.
3. **Logging and Debugging:** Implement logging of kernel messages using printk() for debugging and monitoring module activity.
4. **Dynamic Loading and Unloading:** Enable dynamic loading (insmod) and unloading (rmmod) of the kernel module.

**System Calls and APIs to be Used**:

1. **Character Device Management:** register\_chrdev(), unregister\_chrdev()
2. **Memory Allocation:** kmalloc(), kfree()
3. **File Operations:** file\_operations structure for open, read, write, and release operations
4. **Kernel Logging:** printk()

**Implementation Steps**:

**1. Environment Setup:**

* Set up a Linux environment with kernel headers and development tools.

**2. Basic Module Creation:**

* Write a simple "Hello World" kernel module to test the environment.

**3. Virtual Device Implementation:**

* Register the character device and implement file operations.
* Manage read and write operations with a circular buffer for data storage**.**
  1. **Testing and Debugging:**
* Test the module with custom user-space programs.
* Debug using kernel logs and error handling.
  1. **Documentation:**
* Document the design, implementation steps, and usage of the kernel module.

**Expected Outcomes**:

* 1. A custom kernel module that registers a virtual character device and allows user interaction.
  2. A deeper understanding of kernel architecture, device drivers, and memory management.
  3. Practical experience in kernel module development and debugging.

**Tools and Platform**:

1. **Development**: GCC, Makefile
2. **Platform**: Linux with kernel development headers installed
3. **Tools**: dmesg for debugging, insmod and rmmod for module management

**Conclusion**:

This project will provide hands-on experience with kernel development and OS-level programming. It will enhance understanding of how operating systems handle device management and enable students to explore advanced concepts in kernel programming.