Multi-Process File System

Project Description:

The goal of this project is to design and implement a multi-process file system that provides file sharing and memory management functionalities. The file system should support concurrent access by multiple processes, handle file permissions, and manage memory efficiently.

Project Components:

1. File System Structure: This component involves designing and implementing the structure of the file system. It includes creating a root directory to organize files, a file allocation table to track file locations, and data blocks to store actual file data. Each file will have its own metadata, such as size, permissions, and the location of its data blocks.
2. IPC Mechanism: The Inter-Process Communication (IPC) mechanism allows processes to communicate and share data with each other. You can choose from different IPC techniques like shared memory, message passing, or pipes to enable processes to exchange information and collaborate on accessing shared files.
3. Process Management: The process management module handles process creation, termination, and scheduling. It ensures that processes have appropriate access to shared files and enforces file permissions. This component plays a crucial role in coordinating the activities of multiple processes accessing the file system concurrently.
4. Memory Management: Efficient memory management is essential for both the file system and the processes running on it. This component involves implementing a memory management system that allocates and deallocates memory effectively. Techniques such as paging or segmentation can be employed to manage memory efficiently.
5. File Operations: Basic file operations like creating, opening, reading, writing, and deleting files need to be implemented. These operations should be designed to allow multiple processes to perform them concurrently while maintaining data integrity and consistency.
6. File Permissions: Implementing a permission system allows processes to control access to their files. This component involves implementing read, write, and execute permissions for each file. The file system should enforce these permissions during file operations to ensure proper security and access control.
7. Error Handling: Developing a robust error handling mechanism is crucial to handle various error conditions that may arise during file system operations. Examples of such errors include file not found, insufficient permissions, or out-of-memory errors. Proper error handling improves the reliability and stability of the file system.
8. Testing and Evaluation: Creating a comprehensive set of test cases is essential to validate the functionality and correctness of the file system. Test scenarios should include concurrent access, file sharing, permission enforcement, and memory management. Thorough testing helps identify and resolve any issues or bugs in the implementation.

Optional Enhancements:

- Implement directory support for organizing files into hierarchical structures.
- Develop a user interface to interact with the file system, allowing users to perform file operations and manage processes.
- Implement file caching mechanisms to improve read/write performance.
- Integrate file system encryption and decryption for secure file storage.