**CSE 344 - Homework #1**

**Secure File and Directory Management System Report**  
**Student Name:** Anhelina Bondarenko  
**Student ID:** 220104004928

**1. Introduction**

This project is a **Secure File and Directory Management System** implemented in **C** using **Linux system calls**. The system allows users to:

* Create and delete files and directories.
* List files and directories.
* Read and append to files using file locks.
* Maintain a log of all operations.
* Display logs of executed commands.
* Provide a user-friendly interface with command-line options.

**2. Code Explanation**

The implementation consists of three main files:

**2.1 fileManager.h**

This header file contains function prototypes for all operations.

**2.2 fileManager.c**

This file implements all **core functionalities**:

* **createDir(folderName, logFile):** Creates a directory if it does not already exist.
* **createFile(fileName, logFile):** Creates a file if it does not already exist.
* **listDir(folderName, logFile):** Uses a child process to list the contents of a directory.
* **listFilesByExtension(folderName, extension, logFile):** Lists files with a specific extension into child process.
* **readFile(fileName, logFile):** Reads the content of a file.
* **appendToFile(fileName, content, logFile):** Appends text to a file with a lock to prevent race conditions.
* **deleteFile(fileName, logFile):** Uses a child process to delete a file.
* **deleteDir(folderName, logFile):** Uses a child process to delete an empty directory.
* **showLogs(logFile):** Displays logs of executed operations.

**Helper Functions:**

* **saveLogs(format, logFile, ...):** Saves formatted log messages with timestamps.
* **getCurrentTimestamp(buffer, size):** Retrieves the current timestamp in [YYYY-MM-DD HH:MM:SS] format.

**2.3 main.c**

This file contains the main function, which handles user input and executes commands.

**Main Function Workflow:**

1. **Argument Handling:**
   * If no arguments are provided, a usage guide is displayed, and user input is taken via fgets().
   * If arguments are provided, they are parsed from argv.
2. **Command Execution:**
   * The program opens log.txt with appropriate permissions.
   * A do-while loop continuously reads user input and processes commands.
   * Inside the loop, strcmp() is used to match user commands and call corresponding functions.
3. **Process Management:**
   * Certain commands (listDir, listFilesByExtension, deleteFile, deleteDir) use fork() to create child processes.
   * The parent process waits for the child to complete using wait(NULL).
4. **Exit Condition:**
   * The program continues until the user enters exit.
   * close(logFile) ensures proper resource cleanup before termination.

**3. Screenshots**

Screenshots represent the following test scenarios (extended list from assignment):

1. Creating a directory (testDir).
2. Creating a file (example.txt) and writing "Hello, World!".
3. Creating files text.txt and prog.c into testDir
4. Listing files in testDir.
5. Reading example.txt and appending "New Line".
6. Deleting example.txt and displaying logs.
7. Listing files by extension “.c” in testDir
8. Deleting files text.txt and prog.c from testDir
9. Deleting testDir
10. Displaying logs

Зображення, що містить текст, знімок екрана, програмне забезпечення, монітор

Вміст, створений ШІ, може бути неправильним.

Зображення, що містить текст, знімок екрана, документ

Вміст, створений ШІ, може бути неправильним.Зображення, що містить текст, знімок екрана, Шрифт, документ

Вміст, створений ШІ, може бути неправильним.

**4. Conclusion**

**Challenges Faced:**

* Implementing file locks to prevent simultaneous writes.
* Handling process creation (fork) correctly.
* Ensuring proper logging of all operations.

**Solutions Implemented:**

* Used fcntl() for file locking in appendToFile().
* Used fork() and wait() for process management.
* Created a log.txt file to store operation history, file descriptor passed to command functions for further writing

**Final Thoughts:**

This project successfully implements a secure and efficient file management system in C, following Linux system calls and process management techniques. The logging mechanism provides auditability, and the program supports robust error handling.