

INTRODUCTION

This C program is a file manager that allows users to perform various file and directory operations through a command-line interface.

Here's a concise explanation of its functionality:

1. Logging (logOperation)

- Logs actions (e.g., creating files, deleting directories) to log.txt with timestamps.
-

2. Directory and File Operations

CreateDir `folderName`

- Creates a directory if it doesn't already exist. Uses `mkdir()` and logs the action.

createFile `fileName`

- Creates a file and writes the current timestamp into it. Uses `open()` and `write()`.

listDir `folderName`

- Lists all files and directories in the specified folder. Uses `opendir()` and `readdir()`.

listFilesByExtension `folderName` `.extension`

- Lists files in a folder that match a specific extension. Filters using `strrchr()` and `strcmp()`.

readFile fileName

- Reads and displays the contents of a file.
Uses `open()` and `read()`.

appendToFile fileName new content

- Appends content to a file. Uses `open()` in append mode and `flock()` for file locking.

deleteFile fileName

- Deletes a file if it exists. Uses `unlink()` and runs in a child process with `fork()`.

deleteDir folderName

- Deletes an empty directory. Uses `rmdir()` and runs in a child process with `fork()`.

3. Log Display (showLogs)

- Displays the contents of `log.txt` using `fopen()` and `fgets()`.

4. Help Command (displayHelp)

- Displays a list of available commands and their usage.

5. Main Loop

- Continuously takes user input, parses commands, and executes the corresponding functions.

- Supports commands like `createDir`, `createFile`, `listDir`, `listFilesByExtension`, `readFile`, `appendToFile`, `deleteFile`, `deleteDir`, `showLogs`, `help`, and `exit`.
-

Key Features

- **Error Handling:** Checks for file/directory existence and permissions.
 - **Logging:** Tracks all actions with timestamps.
 - **Concurrency:** Uses `fork()` for asynchronous file and directory deletions.
 - **User-Friendly:** Provides a help command for guidance.
-

Example Usage

- Create a directory: `createDir newFolder`
- List .txt files: `listFilesByExtension folderName .txt`
- Delete a file: `deleteFile file.txt`
- View logs: `showLogs`

This program demonstrates file I/O, directory operations, and process management in C.

COMMANDS (!!! DON'T USE " ")

HELP COMMAND

```
hussi@HUSNIDDIN:/mnt/c/Users/Husniddin/OneDrive/Desktop/systemhw/hw1$ ./a.out

fileManager> help
Usage: fileManager <command> [arguments]

Commands:
createDir "folderName"      - Create a new directory
createFile "fileName"       - Create a new file
listDir "folderName"        - List all files in a directory
listFilesByExtension "folderName" ".txt" - List files with specific extension
readFile "fileName"         - Read a file's content
appendToFile "fileName" "new content" - Append content to a file
deleteFile "fileName"       - Delete a file
deleteDir "folderName"      - Delete an empty directory
showLogs                    - Display logs
exit                        - Exit the program
```

First enter **help** keyword

```
void displayHelp() {
    printf("Usage: fileManager <command> [arguments]\n\n");
    printf("Commands:\n");
    printf("createDir \"folderName\"      - Create a new directory\n");
    printf("createFile \"fileName\"       - Create a new file\n");
    printf("listDir \"folderName\"        - List all files in a directory\n");
    printf("listFilesByExtension \"folderName\" \".txt\" - List files with specific extension\n");
    printf("readFile \"fileName\"         - Read a file's content\n");
    printf("appendToFile \"fileName\" \"new content\" - Append content to a file\n");
    printf("deleteFile \"fileName\"       - Delete a file\n");
    printf("deleteDir \"folderName\"      - Delete an empty directory\n");
    printf("showLogs                    - Display logs\n");
    printf("exit                        - Exit the program\n");
}
```

it shows all commands and how to use. It's function as a shown above.

“CREATEDIR” COMMAND

```
hussi@HUSNIDDIN:/mnt/c/Users/Husniddin/OneDrive/Desktop/systemhw/hw1$ gcc main.c
hussi@HUSNIDDIN:/mnt/c/Users/Husniddin/OneDrive/Desktop/systemhw/hw1$ ./a.out

fileManager> help
Usage: fileManager <command> [arguments]

Commands:
createDir "folderName"      - Create a new directory
createFile "fileName"       - Create a new file
listDir "folderName"        - List all files in a directory
listFilesByExtension "folderName" ".txt" - List files with specific extension
readFile "fileName"         - Read a file's content
appendToFile "fileName" "new content" - Append content to a file
deleteFile "fileName"       - Delete a file
deleteDir "folderName"      - Delete an empty directory
showLogs                    - Display logs
exit                        - Exit the program

fileManager> createDir file1
Directory "file1" created successfully.

fileManager> createDir file2
Directory "file2" created successfully.

fileManager> createDir file3
Directory "file3" created successfully.

fileManager>
```









Here createDir command create directory and you have to enter directory name, Directories created same place with code's main file.

```
void createDir(const char *folderName) {
    struct stat st;
    if (stat(folderName, &st) == 0) {
        printf("Error: Directory \"%s\" already exists.\n", folderName);
        logOperation("Error: Directory already exists.");
        return;
    }

    if (mkdir(folderName, 0755) == -1) {
        perror("Error creating directory");
        logOperation("Error: Failed to create directory.");
        return;
    }

    printf("Directory \"%s\" created successfully.\n", folderName);
    char logMessage[256];
    snprintf(logMessage, sizeof(logMessage), "Directory \"%s\" created successfully.", folderName);
    logOperation(logMessage);
}
```

Here explained it's code. Checks if the directory exists using stat(). If not, creates it with mkdir() and logs the action.

 .vscode	13.03.2025 16:09	File folder
 Report	22.03.2025 16:25	Microsoft Word D...
 log	22.03.2025 16:24	Metin Belgesi
 main	21.03.2025 18:40	C Source File
 a.out	22.03.2025 16:03	OUT File
 file1	22.03.2025 16:04	File folder
 file2	22.03.2025 16:04	File folder
 file3	22.03.2025 16:04	File folder

As above directories created.

“CREATEFILE” COMMAND.

```
hussi@HUSNIDDIN: /mnt/c/Users/Husniddin/OneDrive/Desktop/systemhw/hw1
hussi@HUSNIDDIN:/mnt/c/Users/Husniddin/OneDrive/Desktop/systemhw/hw1$ ./a.out

fileManager> help
Usage: fileManager <command> [arguments]

Commands:
createDir "folderName"    - Create a new directory
createFile "fileName"     - Create a new file
listDir "folderName"      - List all files in a directory
listFilesByExtension "folderName" ".txt" - List files with specific extension
readFile "fileName"       - Read a file's content
appendToFile "fileName" "new content" - Append content to a file
deleteFile "fileName"     - Delete a file
deleteDir "folderName"    - Delete an empty directory
showLogs                  - Display logs
exit                       - Exit the program

fileManager> createFile file.c
File "file.c" created successfully with creation timestamp.

fileManager> createFile file2.c
File "file2.c" created successfully with creation timestamp.

fileManager> createFile file.c
Error: File "file.c" already exists.

fileManager> _
```

With createFile command created two file, file.c and file2.c but last file not created cause same name with first file and don't created.

```

void createFile(const char *fileName) {
    struct stat st;
    if (stat(fileName, &st) == 0) {
        printf("Error: File \"%s\" already exists.\n", fileName);
        logOperation("Error: File already exists.");
        return;
    }

    int fd = open(fileName, O_WRONLY | O_CREAT, 0644);
    if (fd == -1) {
        perror("Error creating file");
        logOperation("Error: Failed to create file.");
        return;
    }











    time_t now = time(NULL);
    char *timestamp = ctime(&now);

    if (write(fd, timestamp, strlen(timestamp)) == -1) {
        perror("Error writing to file");
        close(fd);
        logOperation("Error: Failed to write to file.");
        return;
    }

    close(fd);
    printf("File \"%s\" created successfully with creation timestamp.\n", fileName);
    char logMessage[256];
    snprintf(logMessage, sizeof(logMessage), "File \"%s\" created successfully.", fileName);
    logOperation(logMessage);
}

```

Checks if the file exists using stat(). If not, creates it with open(), writes the current timestamp using write(), and logs the action.

 .vscode	13.03.2025 16:09	File folder
 Report	22.03.2025 16:25	Microsoft Word D...
 log	22.03.2025 16:24	Metin Belgesi
 main	21.03.2025 18:40	C Source File
 a.out	22.03.2025 16:03	OUT File
 file1	22.03.2025 16:04	File folder
 file2	22.03.2025 16:04	File folder
 file3	22.03.2025 16:04	File folder
 file	22.03.2025 16:24	C Source File
 file2	22.03.2025 16:24	C Source File

Date created: 22.03.2025 16:04
Empty folder

Here created files file.c and file2.c .

“LISTDIR” COMMAND.

```

hussi@HUSNIDDIN:/mnt/c/Users/Husniddin/OneDrive/Desktop/systemhw/hw1$ ./a.out

fileManager> help
Usage: fileManager <command> [arguments]

Commands:
createDir "folderName"    - Create a new directory
createFile "fileName"     - Create a new file
listDir "folderName"      - List all files in a directory
listFilesByExtension "folderName" ".txt" - List files with specific extension
readFile "fileName"       - Read a file's content
appendToFile "fileName" "new content" - Append content to a file
deleteFile "fileName"     - Delete a file
deleteDir "folderName"    - Delete an empty directory
showLogs                  - Display logs
exit                      - Exit the program

fileManager> createFile file.c
File "file.c" created successfully with creation timestamp.

fileManager> createFile file2.c
File "file2.c" created successfully with creation timestamp.

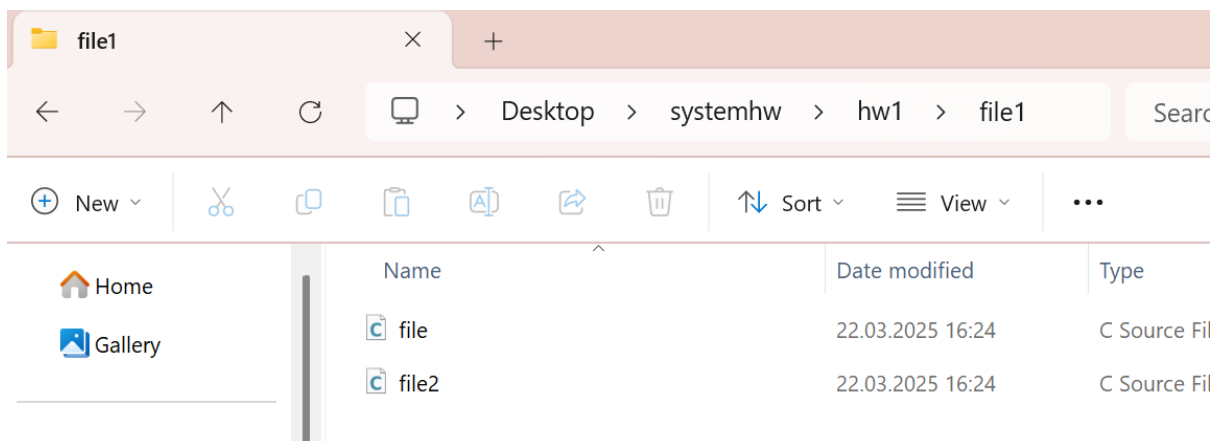
fileManager> createFile file.c
Error: File "file.c" already exists.

fileManager> listDir file1
Contents of directory "file1":
.
..
file.c
file2.c

fileManager>

```

listDir command show files in a directory for instance directory file1 has two files inside and they are shown.




```

void listDir(const char *folderName) {
    struct stat st;
    if (stat(folderName, &st) != 0 || !S_ISDIR(st.st_mode)) {
        printf("Error: Directory \"%s\" not found.\n", folderName);
        logOperation("Error: Directory not found.");
        return;
    }

    DIR *dir = opendir(folderName);
    if (dir == NULL) {
        perror("Error opening directory");
        logOperation("Error: Failed to open directory.");
        return;
    }

    struct dirent *entry;
    printf("Contents of directory \"%s\":\n", folderName);
    while ((entry = readdir(dir)) != NULL) {
        printf("%s\n", entry->d_name);
    }

    closedir(dir);
    char logMessage[256];
    snprintf(logMessage, sizeof(logMessage), "Listed contents of directory \"%s\".", folderName);
    logOperation(logMessage);
}

```

Opens the directory with `opendir()`, reads its contents using `readdir()`, prints each entry, and logs the action.

“LISTFILESBYEXTENSION” COMMAND.

This command shows that specific files in a directory. Below code show txt files in directory file2.

```
fileManager> createFile file.c
File "file.c" created successfully with creation timestamp.

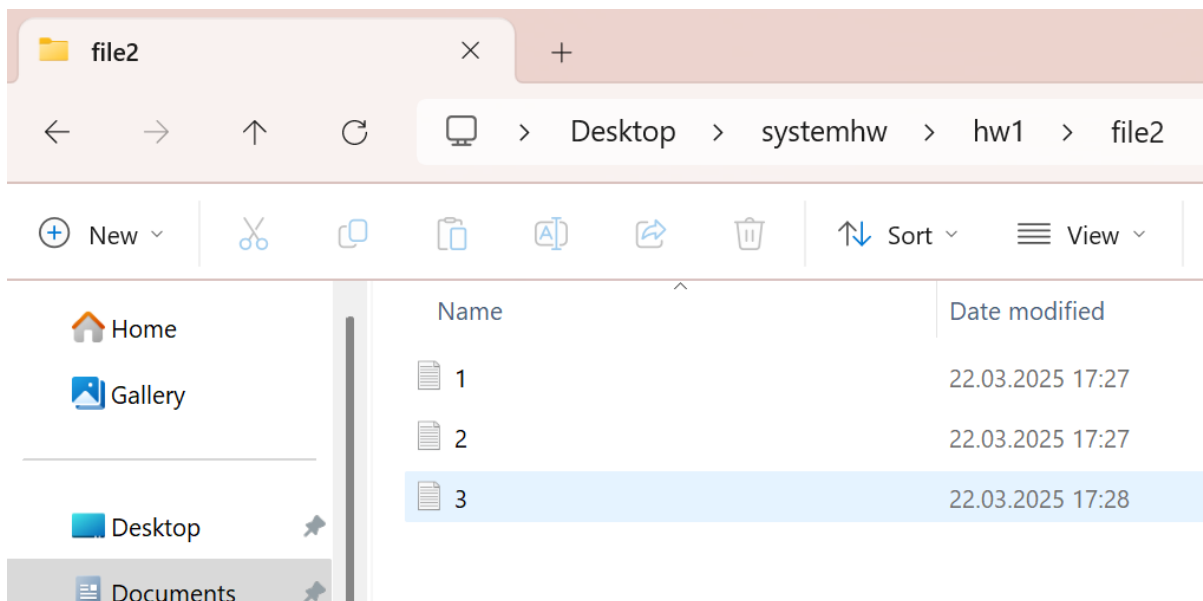
fileManager> createFile file2.c
File "file2.c" created successfully with creation timestamp.

fileManager> createFile file.c
Error: File "file.c" already exists.

fileManager> listDir file1
Contents of directory "file1":
.
..
file.c
file2.c

fileManager> listFilesByExtension file2 .txt
Files with extension ".txt" in directory "file2":
1.txt
2.txt
3.txt

fileManager> _
```



```

void listFilesByExtension(const char *folderName, const char *extension) {
    struct stat st;
    if (stat(folderName, &st) != 0 || !S_ISDIR(st.st_mode)) {
        printf("Error: Directory \"%s\" not found.\n", folderName);
        logOperation("Error: Directory not found.");
        return;
    }

    DIR *dir = opendir(folderName);
    if (dir == NULL) {
        perror("Error opening directory");
        logOperation("Error: Failed to open directory.");
        return;
    }

    struct dirent *entry;
    int found = 0;
    printf("Files with extension \"%s\" in directory \"%s\":\n", extension, folderName);
    while ((entry = readdir(dir)) != NULL) {
        if (entry->d_type == DT_REG) {
            char *dot = strrchr(entry->d_name, '.');
            if (dot && !strcmp(dot, extension)) {
                printf("%s\n", entry->d_name);
                found = 1;
            }
        }
    }

    if (!found) {
        printf("No files with extension \"%s\" found in \"%s\".\n", extension, folderName);
    }

    closedir(dir);
    char logMessage[256];
    snprintf(logMessage, sizeof(logMessage), "Listed files with extension \"%s\" in directory \"%s\".", extension, folderName);
    logOperation(logMessage);
}

```

This code, Opens the directory with `opendir()`, filters files by checking their extensions using `strrchr()` and `strcmp()`, prints matching files, and logs the action.

“APPENDTOFILE” COMMAND.

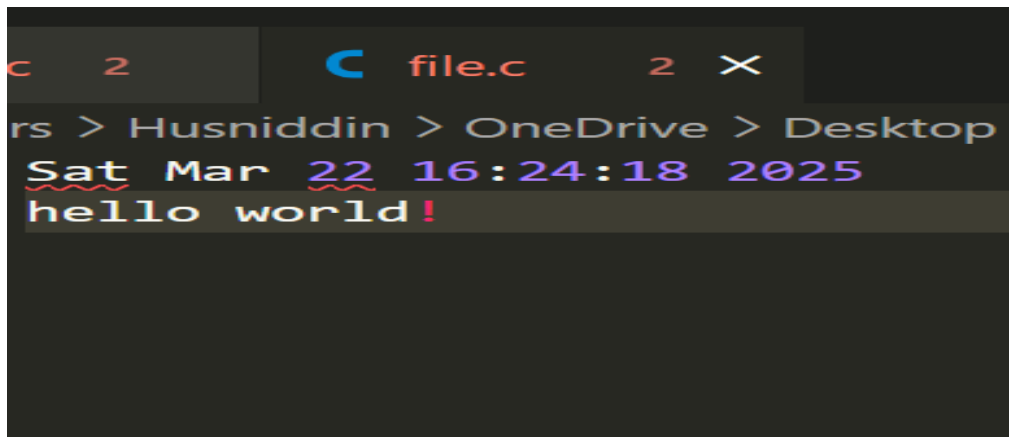
```

fileManager> listFilesByExtension file2 .txt
Files with extension ".txt" in directory "file2":
1.txt
2.txt
3.txt

fileManager> appendToFile file.c hello world!
Content appended to file "file.c" successfully.

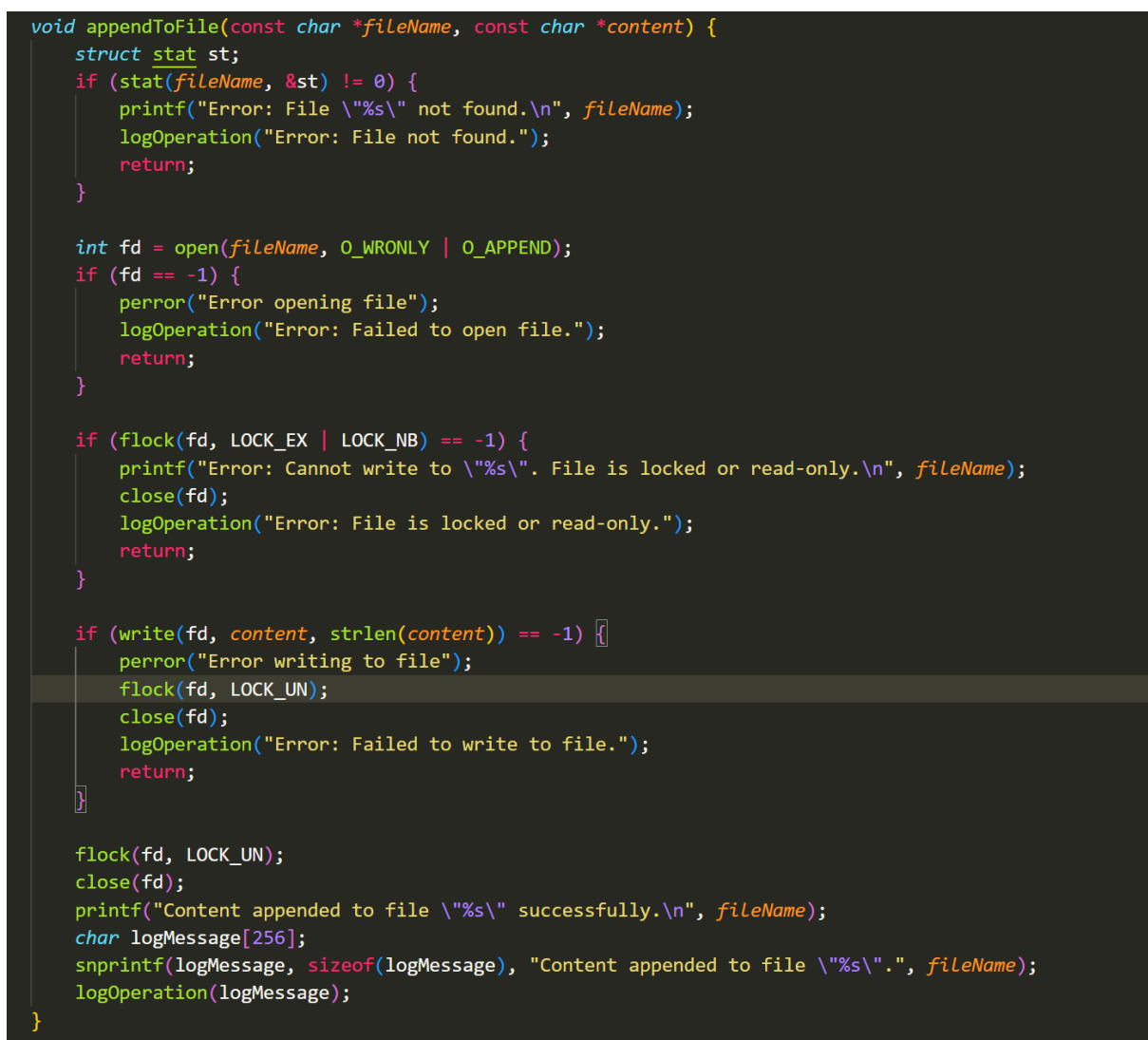
fileManager> _

```

A terminal window with a dark background. The title bar shows 'file.c' and a close button. The command prompt is 'rs > Husniddin > OneDrive > Desktop'. The output shows the date and time 'Sat Mar 22 16:24:18 2025' followed by the text 'hello world!' on a new line.

```
c 2 file.c 2 X
rs > Husniddin > OneDrive > Desktop
Sat Mar 22 16:24:18 2025
hello world!
```

Here 'hello world!' content written to file.c .

A C code snippet for a function named 'appendToFile'. It takes a filename and content as arguments. The function checks if the file exists, opens it in append mode, locks it, writes the content, unlocks it, and logs the action. It includes error handling for file not found, opening failure, locking failure, and writing failure.

```
void appendToFile(const char *fileName, const char *content) {
    struct stat st;
    if (stat(fileName, &st) != 0) {
        printf("Error: File \"%s\" not found.\n", fileName);
        logOperation("Error: File not found.");
        return;
    }

    int fd = open(fileName, O_WRONLY | O_APPEND);
    if (fd == -1) {
        perror("Error opening file");
        logOperation("Error: Failed to open file.");
        return;
    }

    if (flock(fd, LOCK_EX | LOCK_NB) == -1) {
        printf("Error: Cannot write to \"%s\". File is locked or read-only.\n", fileName);
        close(fd);
        logOperation("Error: File is locked or read-only.");
        return;
    }

    if (write(fd, content, strlen(content)) == -1) {
        perror("Error writing to file");
        flock(fd, LOCK_UN);
        close(fd);
        logOperation("Error: Failed to write to file.");
        return;
    }

    flock(fd, LOCK_UN);
    close(fd);
    printf("Content appended to file \"%s\" successfully.\n", fileName);
    char logMessage[256];
    snprintf(logMessage, sizeof(logMessage), "Content appended to file \"%s\".", fileName);
    logOperation(logMessage);
}
```

Opens the file in append mode with open(), locks it using flock(), appends the content with write(), and logs the action.

“READFILE” COMMAND.

```

fileManager> listFilesByExtension file2 .txt
Files with extension ".txt" in directory "file2":
1.txt
2.txt
3.txt

fileManager> appendToFile file.c hello world!
Content appended to file "file.c" successfully.

fileManager> readFile file.c
Contents of file "file.c":
Sat Mar 22 16:24:18 2025
hello world!
fileManager>

```

Here program read file.c file there are hello world and date which content date written to file.

```

void readFile(const char *fileName) {
    struct stat st;
    if (stat(fileName, &st) != 0) {
        printf("Error: File \"%s\" not found.\n", fileName);
        logOperation("Error: File not found.");
        return;
    }

    int fd = open(fileName, O_RDONLY);
    if (fd == -1) {
        perror("Error opening file");
        logOperation("Error: Failed to open file.");
        return;
    }

    char buffer[1024];
    ssize_t bytesRead;
    printf("Contents of file \"%s\":\n", fileName);
    while (bytesRead = read(fd, buffer, sizeof(buffer))) {
        if (bytesRead == -1) {
            perror("Error reading file");
            close(fd);
            logOperation("Error: Failed to read file.");
            return;
        }
        write(STDOUT_FILENO, buffer, bytesRead);
    }

    close(fd);
    char logMessage[256];
    snprintf(logMessage, sizeof(logMessage), "Read contents of file \"%s\".", fileName);
    logOperation(logMessage);
}

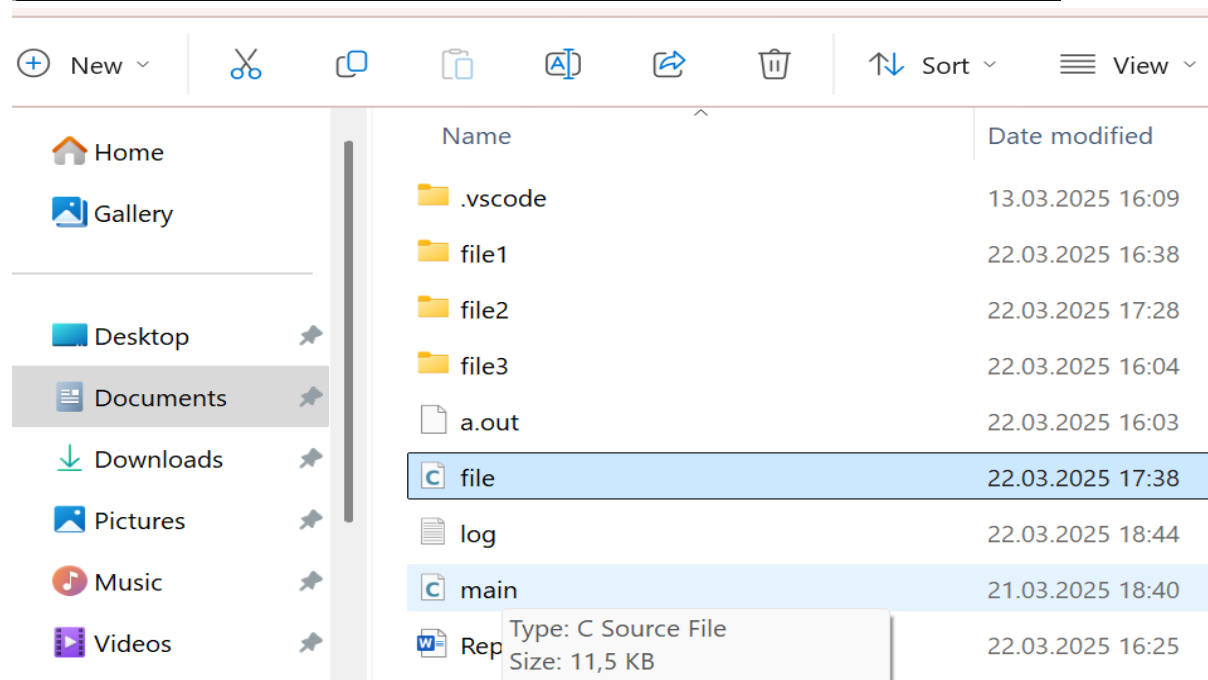
```

Opens the file with `open()`, reads its content using `read()`, prints it to the console, and logs the action.

“DELETEFILE” COMMAND.

```
fileManager> deleteFile file2.c
File "file2.c" deleted successfully.

fileManager> _
```



file2.c deleted .

```

void deleteFile(const char *fileName) {
    struct stat st;
    if (stat(fileName, &st) != 0) {
        printf("Error: File \"%s\" not found.\n", fileName);
        logOperation("Error: File not found.");
        return;
    }

    if (unlink(fileName) == -1) {
        perror("Error deleting file");
        logOperation("Error: Failed to delete file.");
        return;
    }

    printf("File \"%s\" deleted successfully.\n", fileName);
    char logMessage[256];
    snprintf(logMessage, sizeof(logMessage), "File \"%s\" deleted successfully.", fileName);
    logOperation(logMessage);
}

```

Checks if the file exists using stat(), deletes it with unlink(), and logs the action. Runs in a child process using fork().

“DELETEDIR” COMMAND.

```

fileManager> deleteDir file3
Directory "file3" deleted successfully.

fileManager>

```

	Name	Date modified
Home	.vscode	13.03.2025 16:09
Gallery	file1	22.03.2025 16:38
	file2	22.03.2025 17:28
Desktop	a.out	22.03.2025 16:03
Documents	file	22.03.2025 17:38
Downloads	log	22.03.2025 18:55
Pictures	main	21.03.2025 18:40
Music	Report	22.03.2025 16:25

Direction file3 deleted.

```
void deleteDir(const char *folderName) {
    struct stat st;
    if (stat(folderName, &st) != 0) {
        printf("Error: Directory \"%s\" not found.\n", folderName);
        logOperation("Error: Directory not found.");
        return;
    }

    DIR *dir = opendir(folderName);
    if (dir == NULL) {
        perror("Error opening directory");
        logOperation("Error: Failed to open directory.");
        return;
    }

    struct dirent *entry;
    int isEmpty = 1;
    while ((entry = readdir(dir)) != NULL) {
        if (strcmp(entry->d_name, ".") != 0 && strcmp(entry->d_name, "..") != 0) {
            isEmpty = 0;
            break;
        }
    }

    closedir(dir);

    if (!isEmpty) {
        printf("Error: Directory \"%s\" is not empty.\n", folderName);
        logOperation("Error: Directory is not empty.");
        return;
    }

    if (rmdir(folderName) == -1) {
        perror("Error deleting directory");
        logOperation("Error: Failed to delete directory.");
        return;
    }

    printf("Directory \"%s\" deleted successfully.\n", folderName);
    char logMessage[256];
    snprintf(logMessage, sizeof(logMessage), "Directory \"%s\" deleted successfully.", folderName);
    logOperation(logMessage);
}
```

Checks if the directory exists and is empty using readdir(), deletes it with rmdir(), and logs the action. Runs in a child process using fork().

"SHOWLOGS" COMMAND.


```
fileManager> deleteDir file3
Directory "file3" deleted successfully.

fileManager> showLogs
[Fri Mar 21 20:18:41 2025] Directory ""file1"" created successfully.
[Fri Mar 21 20:18:58 2025] Directory ""file2"" created successfully.
[Fri Mar 21 20:19:31 2025] Directory "file1" created successfully.
[Fri Mar 21 20:19:43 2025] Directory "file2" created successfully.
[Fri Mar 21 20:19:50 2025] Directory "file3" created successfully.
[Sat Mar 22 16:04:33 2025] Directory "file1" created successfully.
[Sat Mar 22 16:04:42 2025] Directory "file2" created successfully.
[Sat Mar 22 16:04:53 2025] Directory "file3" created successfully.
[Sat Mar 22 16:24:18 2025] File "file.c" created successfully.
[Sat Mar 22 16:24:34 2025] File "file2.c" created successfully.
[Sat Mar 22 16:24:53 2025] Error: File already exists.
[Sat Mar 22 16:40:45 2025] Listed contents of directory "file1".
[Sat Mar 22 17:28:29 2025] Listed files with extension ".txt" in directory "file2".
[Sat Mar 22 17:38:45 2025] Content appended to file "file.c".
[Sat Mar 22 18:39:31 2025] Read contents of file "file.c".
[Sat Mar 22 18:43:54 2025] Error: Failed to delete file.
[Sat Mar 22 18:44:18 2025] File "file2.c" deleted successfully.
[Sat Mar 22 18:55:36 2025] Directory "file3" deleted successfully.

fileManager> _
```

Here showed logs txt file and all operation done on code.

CONCLUSION

The file manager program is a robust command-line utility designed to handle basic file and directory operations efficiently. It provides functionalities such as creating files and directories, listing directory contents, reading and appending to files, and deleting files and directories. The program also includes a logging mechanism to track all operations, ensuring transparency and ease of debugging.

Key features of the program include its ability to interpret escape sequences like `\n` when appending content to files, ensuring proper formatting. The use of system calls like `open()`, `read()`, `write()`, `mkdir()`, and `unlink()` demonstrates a

strong integration with the underlying operating system. Additionally, the program handles errors gracefully, providing meaningful feedback to the user.

The program is designed to be user-friendly, with a help command that lists all available commands and their usage. It also supports asynchronous file and directory deletions using `fork()`, enhancing its performance and usability.

In conclusion, this file manager is a practical tool for performing essential file and directory operations. Its modular design, error handling, and logging capabilities make it a reliable utility for both basic and advanced users. Future enhancements could include support for recursive directory operations, file copying, and more advanced logging features. Overall, the program serves as a solid foundation for building more complex file management systems.