**File System Implementation Documentation**

**Introduction**

This document provides an overview of the implementation of a simple file system using C++. The system allows users to manipulate directories and files through a set of operations including listing directory contents, creating directories, adding content to files, and reading content from files.

**Implementation Overview**

The file system is implemented using a combination of C++ classes and data structures. The main components include:

1. \*\*File Structure (`struct File`):\*\*

- Represents a file or directory in the file system.

- Has attributes such as `isDirectory` (indicating whether it's a directory), `contents` (a map of directory contents), `name` (name of the file or directory), and `content` (content for files).

2. \*\*File System Class (`class FileSystem`):\*\*

- Manages the overall file system and provides operations to interact with it.

- Includes methods for listing contents (`ls`), creating directories (`mkdir`), adding content to files (`addContentToFile`), and reading content from files (`readContentFromFile`).

**Design Decisions**

1. \*\*File System Structure:\*\*

- The file system is represented as a tree structure with directories as internal nodes and files as leaves.

2. \*\*Path Handling:\*\*

- The `getFile` method uses a stringstream to tokenize and navigate through the file system based on the provided path.

3. \*\*Data Storage:\*\*

- Directories store their contents in a `map` to allow quick access to files and subdirectories.

**Setup Instructions**

To run the file system code, follow these steps:

1. \*\*Clone the Repository:\*\*

```bash

git clone <repository-url>

cd <repository-directory>

```

2. \*\*Compile and Build:\*\*

```bash

g++ -o filesystem filesystem.cpp

```

3. \*\*Run the Code:\*\*

```bash

./filesystem

```

**Usage**

After running the code, you can interact with the file system using the provided methods (`ls`, `mkdir`, `addContentToFile`, `readContentFromFile`). Example usage is demonstrated in the code or can be extended in a separate script.

```cpp

// Example Usage

FileSystem fs;

// Create a directory

fs.mkdir("/root/myFolder");

// List contents of a directory

vector<string> contents = fs.ls("/root");

for (const auto& item : contents) {

cout << item << endl;

}

// Create a file and add content

fs.addContentToFile("/root/myFile.txt", "Hello, World!");

// Read content from a file

string fileContent = fs.readContentFromFile("/root/myFile.txt");

cout << "File Content: " << fileContent << endl;

```

**Conclusion**

This file system implementation provides a basic framework for managing directories and files. Users can extend and customize the code to suit specific requirements. The provided setup script facilitates easy compilation and execution of the code.