**Project 6: File System Manager**

Atuhaire Ambala

Prof. Isac Artzi

CST 315

June 16th 2024

**Project 6: File System Manager**

In this project, we will implement a file system with a hierarchical directory structure, similar to the one in Unix/Linux. This file system will be integrated into a custom shell, allowing users to manage files and directories seamlessly. The hierarchical structure will enable the organization of files in nested directories, providing a clear and logical layout. The shell will support 13 types of commands that can be executed interactively, empowering users to perform various operations such as creating, modifying, deleting, and moving files and directories. This project aims to combine the robust features of a Unix-like file system with an intuitive and powerful command-line interface.

**Responsibilities**

* Atu
  + Code Creation
  + Code testing
  + Documentaiton review
* Ricardo
  + Code review
  + Code testing
  + Documentation

**Methodology / Approach**

The methodology for these enhancements covers several key areas:

1. **Hierarchical Directory Structure:**The shell will implement a hierarchical directory structure similar to Unix/Linux. This structure will allow users to create, rename, move, and delete directories and subdirectories, organizing files in a nested manner. Each directory will maintain a list of its contents, facilitating operations that involve traversing or modifying the directory tree.
2. **File Operations and Management:**The shell will support comprehensive file operations, including creating, renaming, editing, and deleting files. Special attention will be given to file creation, where a flag will specify the amount of random data to generate, eliminating the need for manual file editing. This functionality ensures efficient file management and reduces the overhead of file operations.
3. **Information Retrieval and Display:**The shell will provide commands to retrieve and display both basic and detailed information about files and directories. This includes metadata such as size, permissions, creation and modification dates, and more. Users can utilize flags to request detailed information, ensuring they have access to all necessary data for efficient file system management.
4. **Searching Features:**The shell will include commands to search for files within the directory tree and duplicate files and directories. These features enhance the user’s ability to manage and organize their files, making it easier to locate specific items and create backups or copies of important data. The search functionality will traverse the directory tree recursively, providing results that help users quickly find their target files.
5. **Handling Non-Empty Directories:** When deleting directories, the shell will provide special handling for non-empty directories through a flag in the delete command. This ensures that users are aware of the contents being deleted and can confirm their intention to remove all files and subdirectories within a directory, preventing accidental data loss.

**Assumptions**

Several key assumptions underpin the design and functionality of this shell, ensuring smooth and efficient operation. Firstly, it is assumed that each file and directory name is unique within its respective parent directory. This eliminates conflicts and simplifies operations such as creation, renaming, and deletion, thereby maintaining a clear and organized file structure. When creating files, it is assumed that the user provides a valid size in bytes for random data generation. This prevents errors related to invalid sizes and ensures efficient use of storage and memory resources.

Additionally, the shell assumes that users will provide explicit consent via a special flag when deleting non-empty directories. This protects against accidental data loss by requiring users to confirm their actions before potentially removing valuable data. There is also an assumption of predefined limits on the number of files and subdirectories within any given directory, which aids in efficient memory management and prevents performance degradation or system crashes.

Consistent formatting of directory paths, using forward slashes to separate directories, is another assumption that simplifies path parsing and interpretation. This consistency is crucial for navigating the file system and performing various operations. Furthermore, it is assumed that the underlying file system and hardware provide sufficient resources to handle the shell's operations. This ensures the shell can perform tasks like file creation, random data generation, and directory manipulation without resource constraints.

Finally, the shell assumes that users have a basic understanding of command-line operations and the specific commands supported by the shell. This allows for a more complex design that relies on correct command usage, reducing the need for extensive error handling for incorrect inputs. These assumptions collectively simplify the development process and ensure the shell functions reliably and efficiently within the specified parameters.

**Implementation of commands**

 Create and Rename Directories:

* The shell allows users to create new directories and rename existing ones using simple commands. Creating a directory involves specifying the desired path, and renaming involves providing the current name and the new desired name. These operations update the internal directory structure, ensuring the hierarchical organization is maintained.

 Delete Directories:

* Deleting directories can be done with special handling for non-empty directories through a specific flag. This ensures that users are aware of the contents being deleted, preventing accidental data loss. The deletion process recursively removes all files and subdirectories within the target directory, updating the internal structure accordingly.

 Create, Rename, Edit, and Delete Files:

* Users can manage files comprehensively, including creating new files, renaming them, editing their contents, and deleting them. When creating files, the shell supports a flag to specify the amount of random data to generate, simplifying file initialization. Renaming and deleting files update the directory structure to reflect these changes.

 Random Data Generation for File Creation:

* When a file is created, a special flag allows users to specify the amount of data to be randomly generated and written to the file. This eliminates the need for manual editing, streamlining the file creation process and ensuring that files have meaningful content from the start.

 Move Files Across Directories:

* The shell supports moving files from one directory to another. This involves specifying the source file and the destination directory, updating the internal structure to reflect the new file location. This command ensures that file paths are managed correctly and efficiently.

 Duplicate Files:

* Users can duplicate files within the directory structure. This command copies the contents and metadata of the source file to a new file, updating the directory structure to include the duplicate. This is useful for creating backups or working copies of important files.

 Duplicate Directories:

* The shell also supports duplicating entire directories, including all their contents. This recursive operation ensures that the entire structure and all files within the source directory are copied to a new location, maintaining the hierarchy and content integrity.

 Search for a File in a Directory Tree:

* Users can search for files within the entire directory tree starting from a specified node. The search command traverses the directory structure recursively, matching file names and returning their paths. This helps users quickly locate specific files within large hierarchies.

 Display a Directory Tree:

* The shell can display the directory tree starting from a given node, providing a hierarchical view of directories and files. This command outputs a structured representation of the file system, helping users visualize and navigate their directory structures.

 Get Basic Information About a File:

* Users can retrieve basic information about a file, such as its size, creation date, and last modified date. This command provides essential metadata that helps users understand the file's attributes and status without delving into more detailed properties.

 Get Detailed Information About a File:

* For more in-depth file analysis, the shell can provide detailed information, including permissions, owner, group, and access times. A special flag triggers this detailed output, offering comprehensive insights into the file's properties and usage.

 Get Basic Information About a Directory:

* Similar to files, users can get basic information about directories, such as their size and the number of contained files and subdirectories. This command helps users gauge the contents and significance of directories within the hierarchy.

 Get Detailed Information About a Directory:

* Detailed directory information includes permissions, owner, group, and access times, available through a special flag. This command provides a thorough overview of directory properties, enabling better management and security assessment of directory contents

**Management of the File Directory**

The management of the file directory in this shell involves several key operations: creation, renaming, deletion, moving, duplicating, searching, and retrieving information about files and directories. Each of these operations interacts with the internal data structures that represent the file system. Here’s a detailed description of how these operations are managed

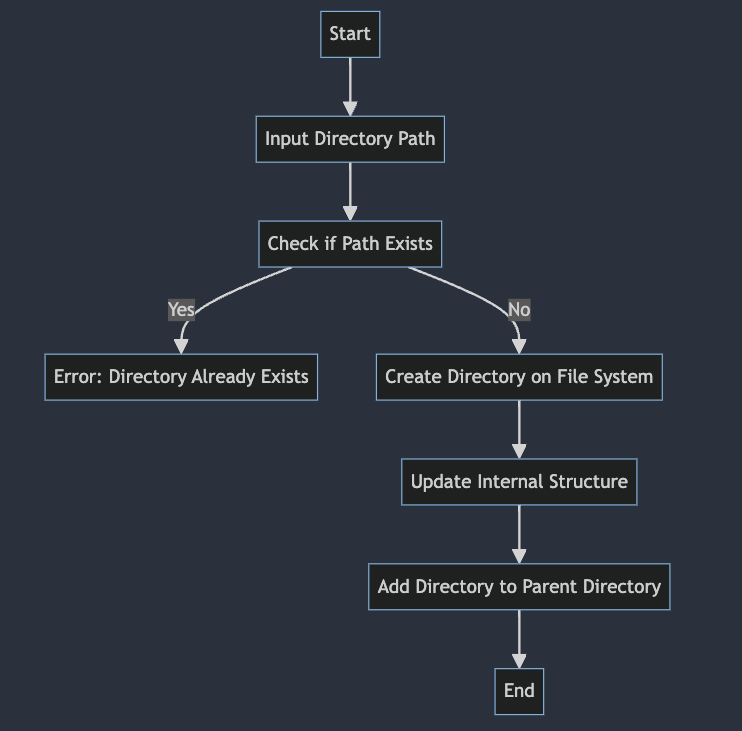
Here is a description of the internal data structures for directories and files

A screen shot of a computer screen

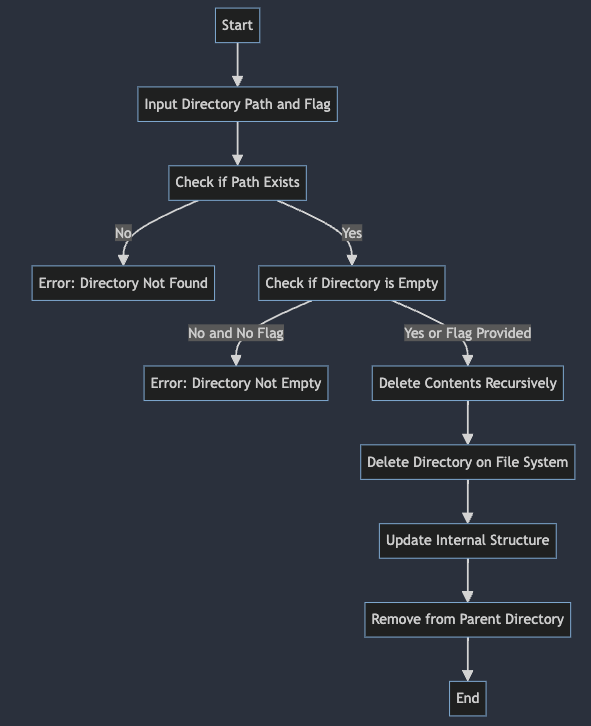
Description automatically generated

Below are key operations in the management of the file directory

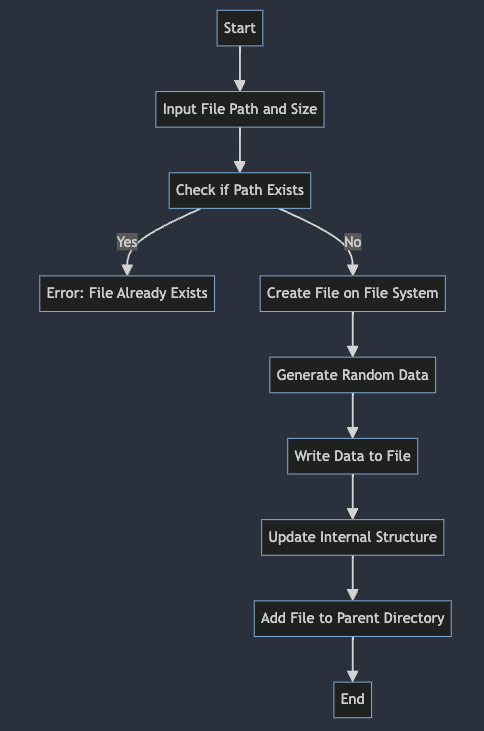
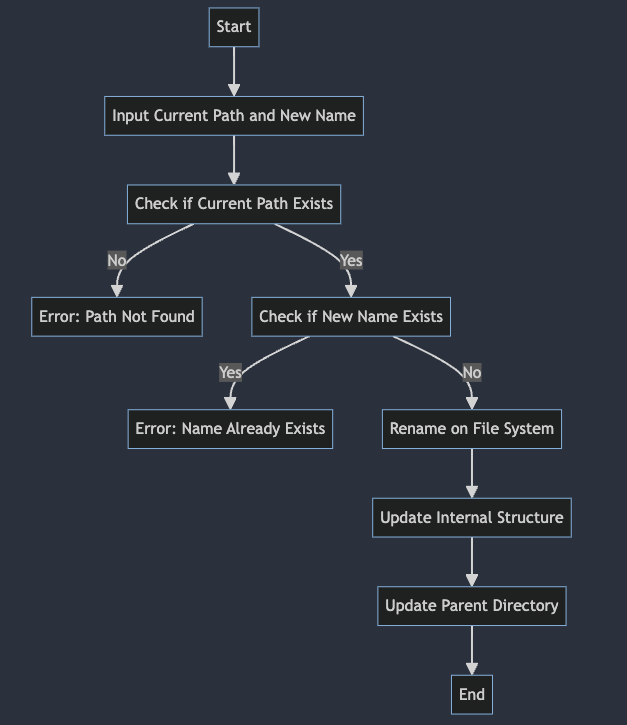
1. **Creating and Renaming Directories:** Users can create new directories by specifying a path, with the shell ensuring the path does not already exist before proceeding. This prevents conflicts and maintains a clean directory structure. When renaming directories, the shell validates the current path and the new name, performs the rename operation, and updates the internal structure accordingly. This ensures that the directory hierarchy remains consistent and up-to-date.



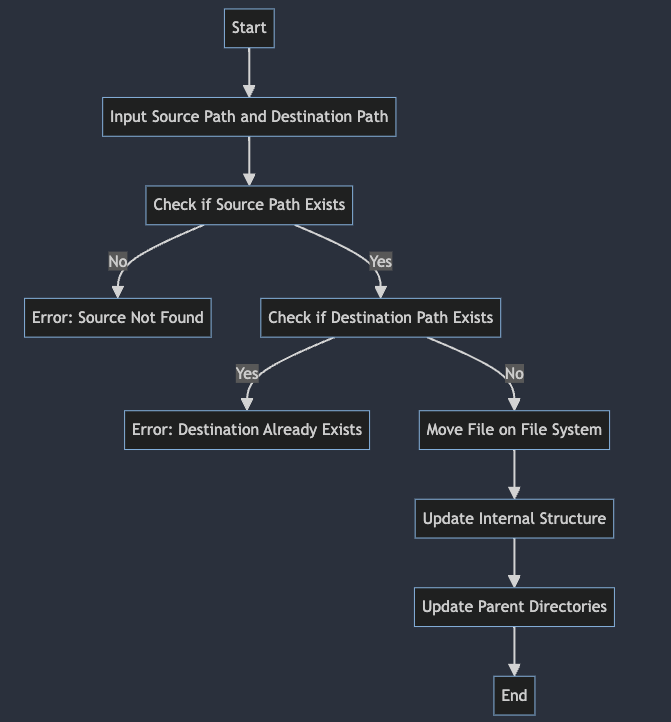
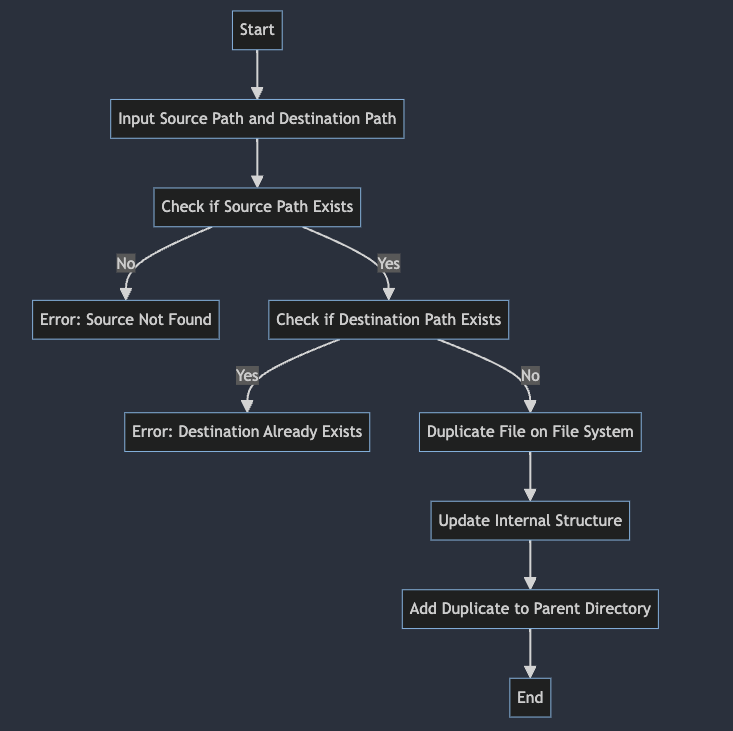
1. **Deleting Directories:** The shell supports the deletion of directories, including non-empty ones, through a special flag. This operation involves recursively deleting all contents within the directory, updating the file system, and modifying the internal structure. This capability allows users to manage their directories efficiently, removing unwanted or obsolete directories while maintaining an accurate directory structure.



1. **Creating, Renaming, Editing, and Deleting Files:** The shell provides comprehensive file management capabilities, including creating, renaming, editing, and deleting files. When creating files, a special flag allows users to specify the amount of random data to generate, simplifying file initialization. Renaming and deleting files involve validating the paths, performing the operations, and updating the internal structure. These functionalities ensure that file operations are handled smoothly, keeping the directory structure accurate and organized.

1. **Moving and Duplicating Files and Directories:** The shell supports moving files across directories by specifying source and destination paths. This operation updates both the file system and internal structure to reflect the new location, ensuring seamless file management. Additionally, duplicating files and directories creates copies of the original items, accurately reflecting the new copies in both the file system and internal structure. This feature is useful for creating backups and managing multiple versions of files or directories.

1. **Searching and Displaying Directory Trees:** Users can search for files within the directory tree, with the shell providing paths to matching files. This functionality is particularly useful for quickly locating specific files within large directory structures. Additionally, displaying a directory tree provides a hierarchical view of directories and files, helping users visualize the organization of their file system. This enhances navigation and overall directory management.

Searching for a file


1. **Retrieving File and Directory Information:** The shell allows users to obtain both basic and detailed information about files and directories, including size, permissions, ownership, and timestamps. Special flags enable the retrieval of detailed metadata, supporting advanced management and security assessments. This comprehensive information helps users understand the attributes and status of their files and directories, facilitating effective management and organization.

In summary, the internal directory structure is dynamically updated to reflect changes made by file and directory operations. This ensures that the shell always has an accurate representation of the file system, facilitating efficient navigation and management. Key operations update the internal structures as follows:

* **Create and Rename Operations:** Update the parent directory to include the new or renamed item.
* **Delete Operations:** Remove the item from the parent directory and update the structure.
* **Move Operations:** Update both the source and destination directories to reflect the move.
* **Duplicate Operations:** Add the duplicate to the parent directory, maintaining the hierarchy.

**Key code related explanations**

1. Directory and File Structures

The file management system revolves around the structures that represent directories and files. These structures are crucial for storing and managing metadata for files and directories, ensuring the shell can efficiently perform operations like creation, renaming, and deletion.

A screen shot of a computer

Description automatically generated

2. Creating Directories

Creating a directory involves ensuring the specified path does not already exist, creating the directory in the file system, and updating the internal structure to reflect the change.

A screenshot of a computer program

Description automatically generated

3. Creating files with Random Data

When creating a file, users can specify a size for random data generation. This simplifies file initialization by populating the file with random content.

A computer screen shot of a program code

Description automatically generated

4.Renaming Files and Directories

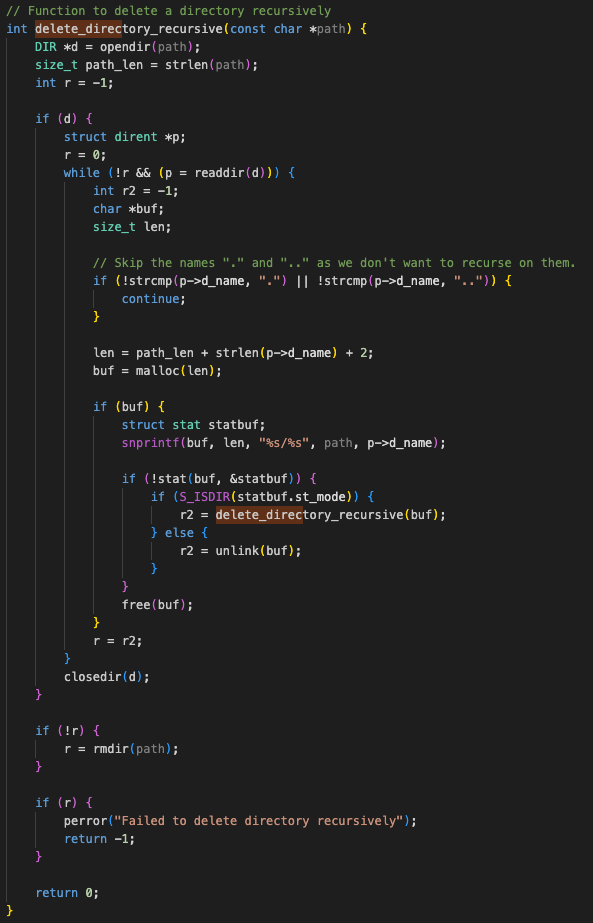
Renaming involves validating the current path and new name, performing the rename operation on the file system, and updating the internal structure.

A computer screen shot of text

Description automatically generated

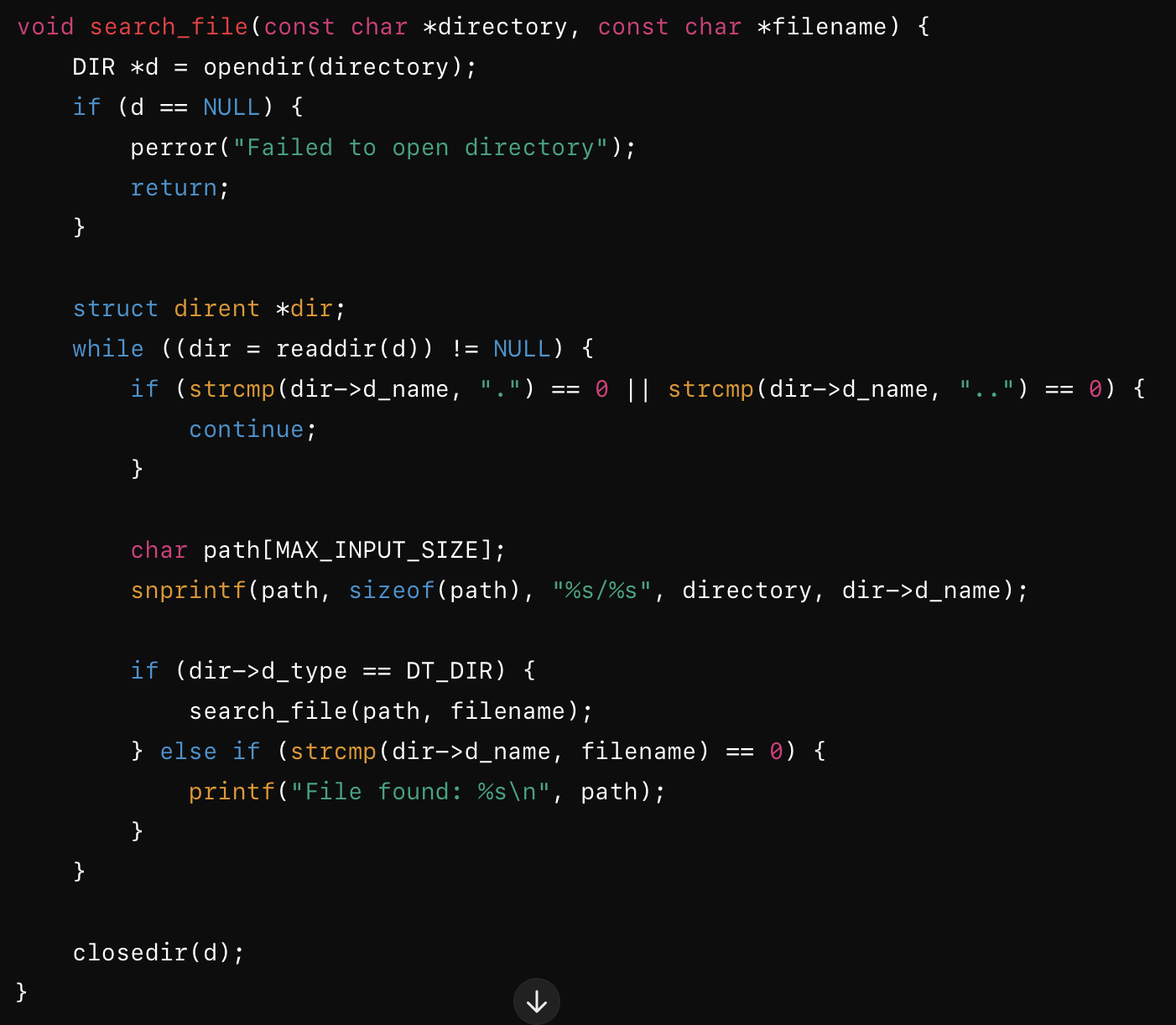
5.Deleting Directory

Deleting a directory involves checking if it is empty or using a special flag to delete non-empty directories. The operation recursively deletes contents and updates the internal structure.



6. Searching for Files

Searching involves traversing the directory tree recursively and matching file names to provide their paths.



7. Displaying Directory Trees

Displaying a directory tree provides a hierarchical view of directories and files, aiding in visualization and navigation.

A screen shot of a computer program

Description automatically generated

These code snippets highlight the key functionalities of the shell's file management system. The operations for creating, renaming, deleting, moving, duplicating, searching, and displaying files and directories are designed to maintain a consistent and organized directory structure, facilitating efficient and intuitive file management for users. The internal structures are dynamically updated to reflect changes, ensuring that the shell always has an accurate representation of the file system

**Testing and Validation**

Testing the creating and renaming of directories. Also tested added a directory when it already exists.

Testing the deletion of directories

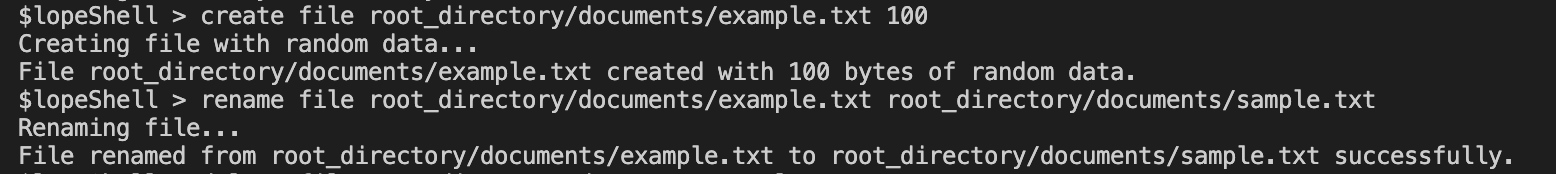
A screenshot of a computer program

Description automatically generated

A black background with blue lines

Description automatically generated

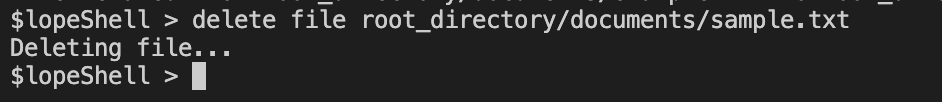
Testing the creation, and renaming of files



A black screen with white text

Description automatically generated

Testing deleting files



Testing the moving of a file, duplicating a file and duplicating a directory

A screen shot of a computer

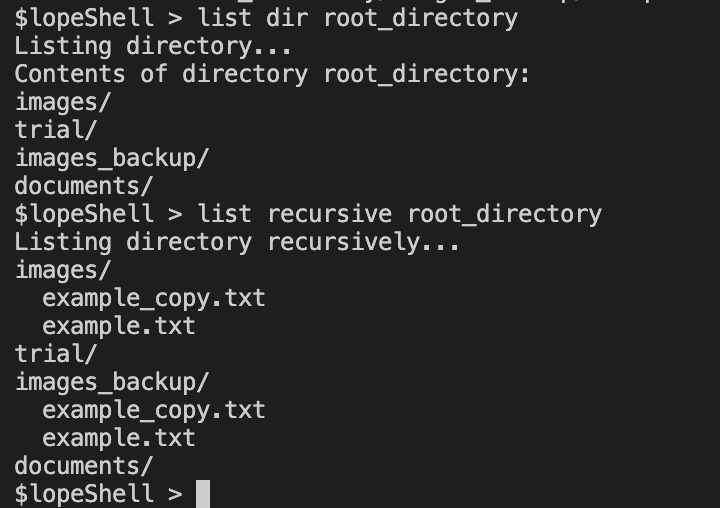
Description automatically generated

Search file testing

A black background with white text

Description automatically generated

Testing directory listing as well as recursive directory listing

 A screenshot of a computer program

Description automatically generated

Testing getting file and directory information, as well getting detailed information for both

A screenshot of a computer

Description automatically generated

**Github Repo**

<https://github.com/AtuAmbala/CST-315/tree/main/Project6%3AFileSystemManager>

**References**

Bach, M. J. (1986). *The Design of the UNIX Operating System*. Prentice Hall.

GNU Project. (n.d.). *GNU C Library Documentation*. Retrieved from https://www.gnu.org/software/libc/manual/

IEEE and The Open Group. (2018). *The Open Group Base Specifications Issue 7, 2018 Edition*. Retrieved from https://pubs.opengroup.org/onlinepubs/9699919799/

Kernighan, B. W., & Ritchie, D. M. (1988). *The C Programming Language* (2nd ed.). Prentice Hall.

Kerrisk, M. (2010). *The Linux Programming Interface: A Linux and UNIX System Programming Handbook*. No Starch Press.

Linux Manual Pages Project. (n.d.). *Linux Man Pages Online*. Retrieved from https://man7.org/linux/man-pages/

Love, R. (2010). *Linux System Programming* (2nd ed.). O'Reilly Media.

Robbins, A., & Beebe, N. (2005). *Classic Shell Scripting*. O'Reilly Media.

Sobell, M. G. (2017). *A Practical Guide to Linux Commands, Editors, and Shell Programming* (4th ed.). Prentice Hall.

Tanenbaum, A. S., & Bos, H. (2015). *Modern Operating Systems* (4th ed.). Pearson.