# Filesystem Data Structures Report

Muhammad Arsalan Hussain mh07607

September 30, 2023

## 1 Introduction

This report describes the data structure used in a filesystem implementation. The data structure comprises a tree/linked-list hybrid to organize files and directories. We will explain the structure and algorithms for traversal and size propagation.

## 2 Data Structures

The filesystem uses several data structures to manage files and directories. These structures include:

### 2.1 Directory Entry Structure

The `dirent` structure represents an entry in a directory. It includes fields such as:

- `name`: The name of the file or directory.
- `namelen`: The length of the entry name.
- `inode`: The index of the associated inode.

### 2.2 Node Structure

The `Node` structure represents a node in the filesystem's tree/linked-list structure. It includes fields for a `dirent` entry, as well as pointers to child, sibling, and parent nodes.

## 3 Traversal Algorithm

The traversal algorithm is used to navigate the filesystem tree structure. The primary function, `traverseUntilParent`, takes a path and traverses the tree until it reaches the parent directory of the target file or directory. The algorithm works as follows:

1. Start at the root node.
2. Iterate through each component of the path, moving to the corresponding child node.
3. Check if the current node exists and has the correct name. If not, return an error.
4. Ensure that the target node is a directory (not a file) if needed.
5. If successful, return 0; otherwise, return -1.

## 4 Size Propagation Algorithm

The size propagation algorithm updates the sizes of parent directories when files or subdirectories are added or removed. It ensures that directory sizes reflect the total size of their contents.

The algorithm works as follows:

1. When creating or deleting a file or directory, calculate its size change.
2. Traverse the tree upward to the root, updating the size of each parent directory by adding or subtracting the size change.
3. Repeat this process until the root is reached.

## 5 Traversal Example

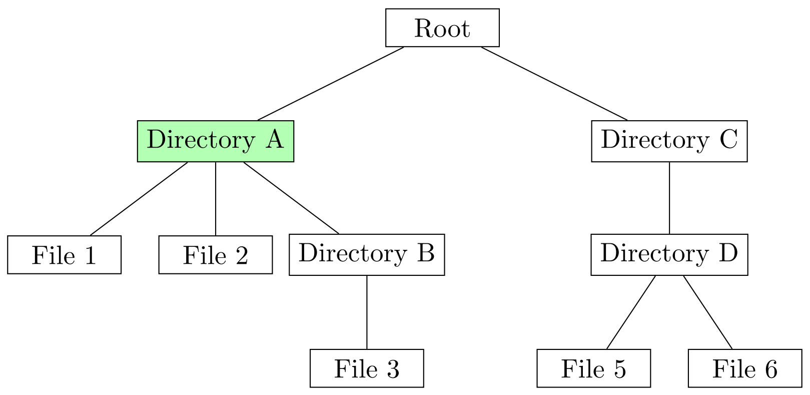

Figure 1: Traversal Example

1. Start at the **Root** node.
2. Move to the first child, **Directory A**.

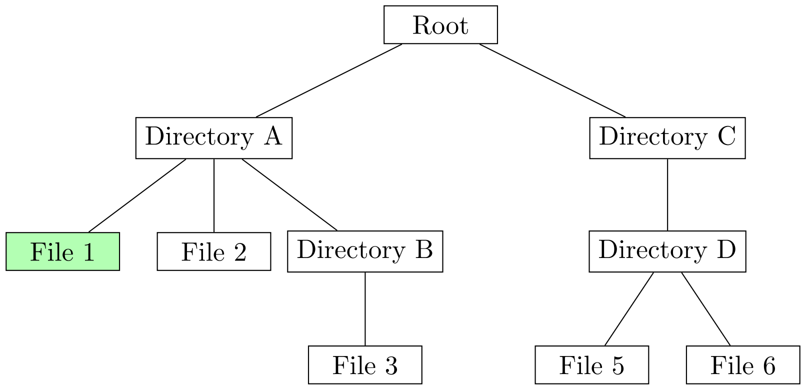

Figure 2: Traversal Example

1. Inside **Directory A**, locate **File 1**.
2. File not found. Move to the next sibling, **File 2**.

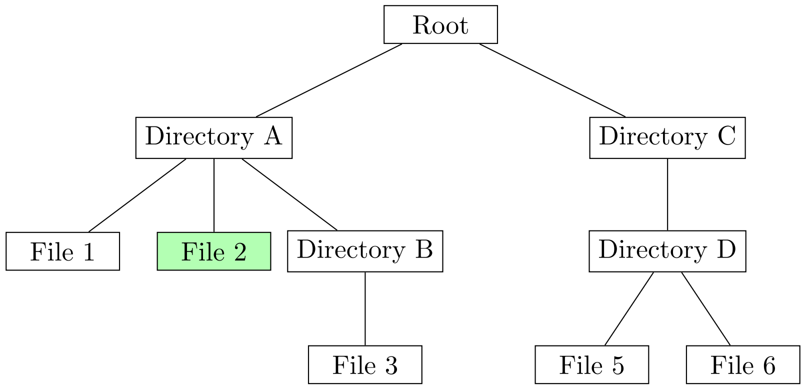

Figure 3: Traversal: Found "File 2"

5. File found! "File 2" has been located within **Directory A**.

## 6 File and Directory Operations

The report briefly describes file and directory operations, such as creating and deleting files and directories. These operations involve manipulating the tree structure, updating inodes and block pointers, and propagating size changes.

## 7 Conclusion

In conclusion, the filesystem implementation utilizes a tree/linked-list hybrid data structure to organize files and directories. Traversal and size propagation algorithms ensure efficient navigation and accurate size information.