Canned FS - File Storage on Student Computer Cluster

Lisa Korver, Eric Xie, Sumatra Dhimoyee, Emre Karabay

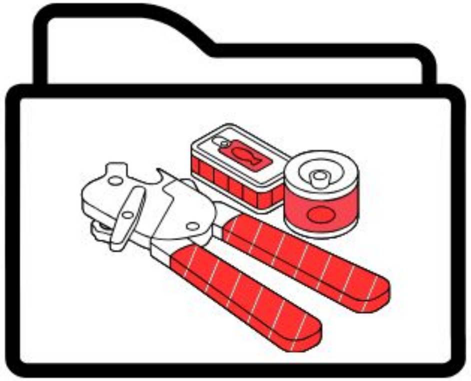
BOSTON UNIVERSITY

Simple Filesystem

Files separated into chunks stored on disk, some files might share chunks

Queries

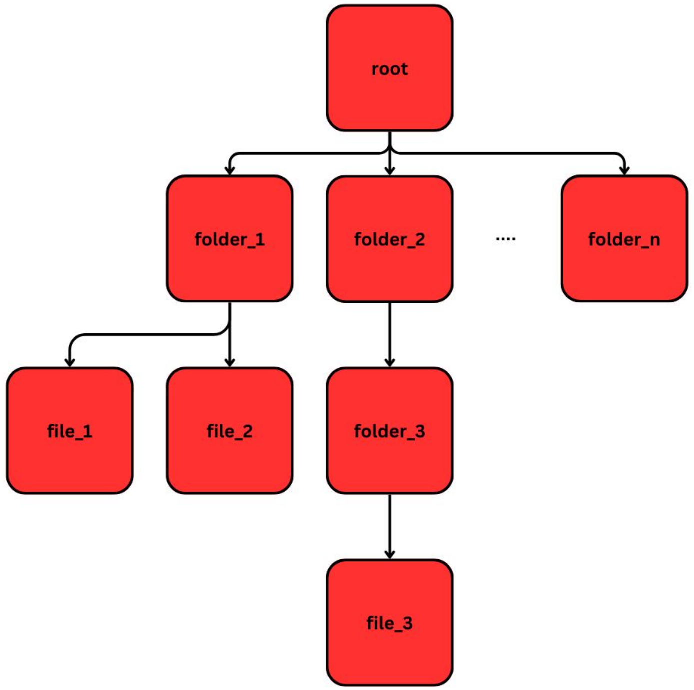
- Create: new file is loaded and added to data structure
- Find: request to download file, chunks are found and returned to user
- Delete: file is removed from system, any chunks deleted
- Update: change data within a file
- List: lists the names of existing files in directory
- Move: change the directory of a file



BOSTON UNIVERSITY

Data Structures for File System

- N-Node Tree for file metadata - indexed by file name
  - Directory location based on tree path
  - Chunk details stored in vector:
    - Start indexes of chunks for each file

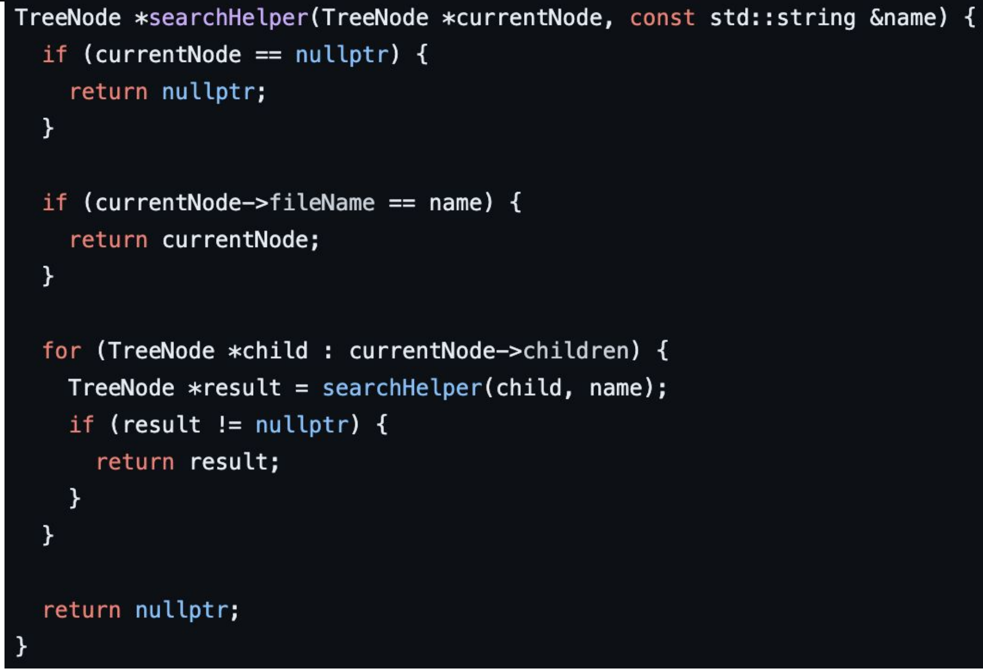


BOSTON UNIVERSITY

Link to canva

Find File

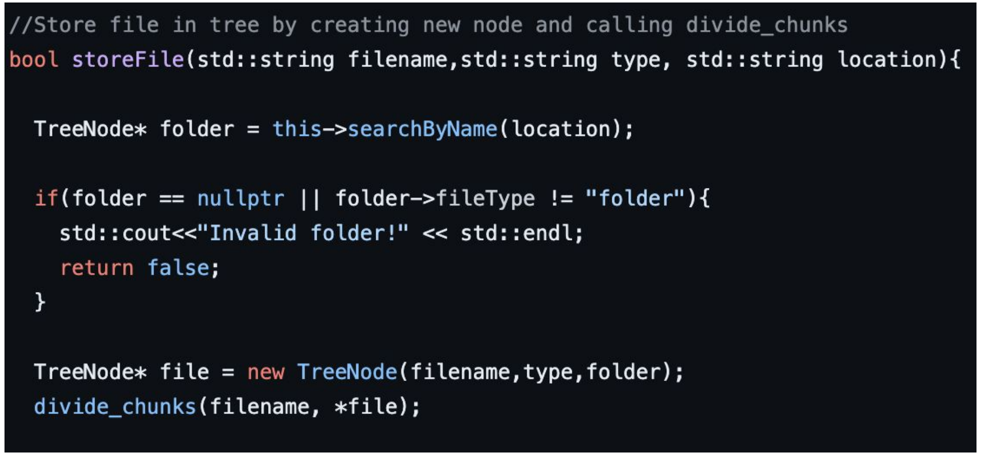
- Implemented with method searchByName
    - Inputs: filename, starting folder
    - Using DFS to search through TreeNodes until file is found
    - Outputs: path to folder
    - O(n)



```
TreeNode *searchHelper(TreeNode *currentNode, const std::string &name) {
  if (currentNode == nullptr) {
    return nullptr;
  }

  if (currentNode->fileName == name) {
    return currentNode;
  }

  for (TreeNode *child : currentNode->children) {
    TreeNode *result = searchHelper(child, name);
    if (result != nullptr) {
      return result;
    }
  }

  return nullptr;
}
```

Create File

- Implemented with method storeFile
    - Inputs: filename, filetype, folder
    - Creates a TreeNode to represent the new file
    - Calls divide_chunks
    - O(n + c)
        - n = # of files in file system
        - c = # of chunks in file



```
//Store file in tree by creating new node and calling divide_chunks
bool storeFile(std::string filename,std::string type, std::string location){

  TreeNode* folder = this->searchByName(location);

  if(folder == nullptr || folder->fileType != "folder"){
    std::cout<<"Invalid folder!" << std::endl;
    return false;
  }

  TreeNode* file = new TreeNode(filename,type,folder);
  divide_chunks(filename, *file);
```

Dividing into Chunks

- Implemented in divide_chunks
    - Inputs: the TreeNode for the given file, the input file name
    - Based on the size of the file, allocates n chunks of 1 kilobyte each and stores their addresses to a vector in the TreeNode

Move

- Implemented with moveFile
  - Inputs: filename, folder name
  - Searches for nodes with filename and folder name
  - Updates parent/children relationships
  - O(n)

Delete

- Implemented with deleteFile
  - Input: file name
  - Finds the node with file name
  - Visit each children/grandchildren with dfs
  - For each visited subfile:
    - Free the storage space, delete Node from Tree

Copy File

- Implemented with copyFile
    - Inputs: original file name, new file name
    - Creates a new TreeNode and copies meta information from original file
    - Copy on Write mechanism: data itself is only copied when one of the files is updated
    - O(n)

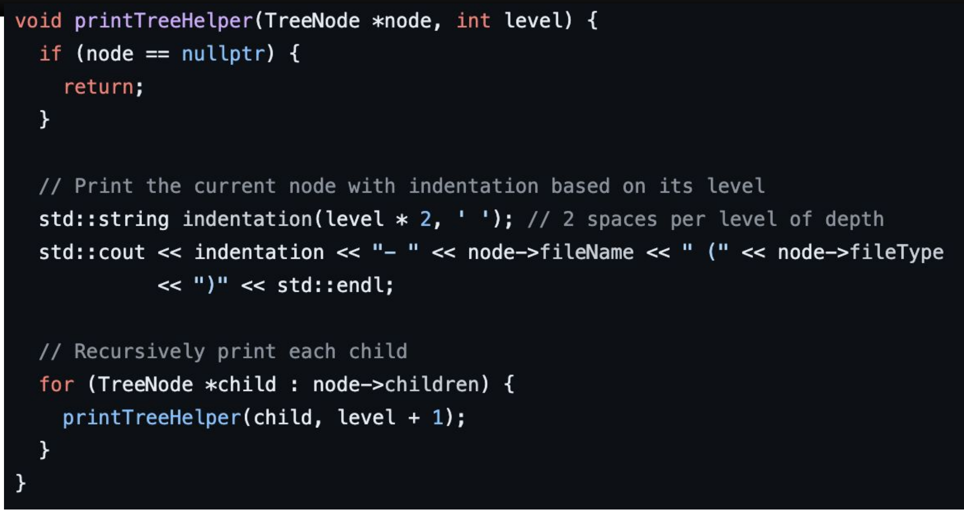
Update

- Implemented with updateFile
  - Inputs: original file name
  - Read all the chunks
  - Update the respective chunks and/ or create new chunks
  - Update the chunk information in the vector
  - O(n+c)

BOSTON UNIVERSITY

List

- Implemented with method printTree that prints all the files at each level of the file
    - Prints each level with indentation
    - O(n)



```
void printTreeHelper(TreeNode *node, int level) {
  if (node == nullptr) {
    return;
  }

  // Print the current node with indentation based on its level
  std::string indentation(level * 2, ' '); // 2 spaces per level of depth
  std::cout << indentation << "- " << node->fileName << " (" << node->fileType
            << ")" << std::endl;

  // Recursively print each child
  for (TreeNode *child : node->children) {
    printTreeHelper(child, level + 1);
  }
}
```

Further Improvements

- Implement command line UI
  - Allow users to navigate through directories
  - Can create/delete from current directory, don’t need to use searchByName which traverses entire tree
  - createFile: O(n+m) -> O(m)
  - updateFile: O(n+m)->O(m)

How to Run on SCC Using Command Line

```
>> ./CannedFS
>> root created @ address:
>> create myFirstFolder folder
>> change_directory myFirstFolder
>> create myFirstFile txt
>> copy myFirstFile mySecondFile
>> create mySecondFolder folder
>> move mySecondFile mySecondFolder
>> find mySecondFile
>> list
```