588. Design In-Memory File System

Description HintsSubmissionsSolutions

- Total Accepted: **951**
- Total Submissions: **2971**
- Difficulty: **Hard**
- Contributors:fallcreek

Design an in-memory file system to simulate the following functions:

ls: Given a path in string format. If it is a file path, return a list that only contains this file's name. If it is a directory path, return the list of file and directory names **in this directory**. Your output (file and directory names together) should in **lexicographic order**.

mkdir: Given a **directory path** that does not exist, you should make a new directory according to the path. If the middle directories in the path don't exist either, you should create them as well. This function has void return type.

addContentToFile: Given a **file path** and **file content** in string format. If the file doesn't exist, you need to create that file containing given content. If the file already exists, you need to **append** given content to original content. This function has void return type.

readContentFromFile: Given a **file path**, return its **content** in string format.

Example:

Input:

```
["FileSystem","ls","mkdir","addContentToFile","ls","readContentFromFile"]
```

[[],["/"],["/a/b/c"],["/a/b/c/d","hello"],["/"],["/a/b/c/d"]]**Output:**

[null,[],null,null,["a"],"hello"]**Explanation:**

Operation	Output	Explanation
FileSystem fs = new FileSystem()	null	The constructor returns nothing.
fs.ls("/")	[]	Initially, directory `/` has nothing. So return empty list.
fs.mkdir("/a/b/c")	null	Create directory `a` in directory `/`. Then create directory `b` in directory `a`. Finally, create directory `c` in directory `b`.
fs.addContentToFile("/a/b/c/d","hello")	null	Create a file named `d` with content `"hello"` in directory `/a/b/c`.
fs.ls("/")	["a"]	Only directory `a` is in directory `/`.
fs.readContentFromFile("/a/b/c/d")	"hello"	Output the file content.

Note:

1. You can assume all file or directory paths are absolute paths which begin with `/` and do not end with `/` except that the path is just `"/"`.
2. You can assume that all operations will be passed valid parameters and users will not attempt to retrieve file content or list a directory or file that does not exist.
3. You can assume that all directory names and file names only contain lower-case letters, and same names won't exist in the same directory.

Subscribe to see which companies asked this question.

```
class FileSystem {
private:
    struct TrieNode {
        bool isFile;
        string content;
```

```
        unordered_map<string, TrieNode *> children;
        TrieNode() : isFile(false) {}
    };

    TrieNode *root;

public:
    FileSystem() {
        root = new TrieNode();
    }

    vector<string> getStrs(string &path) {
        vector<string> ans;
        int i = 1, j = 1;
        while (j <= path.length()) {
            if (i != j && (j == path.length() || path[j] == '/')) {
                ans.push_back(path.substr(i, j - i));
                i = j + 1;
            }
            ++j;
        }
        return ans;
    }

    vector<string> ls(string path) {
        vector<string> strs = getStrs(path);
        TrieNode *curr = root;
        for (string &str : strs)
            curr = curr->children[str];
```

```
    if (curr->isFile)
        return {strs.back()};

    vector<string> ans;
    for (auto &p : curr->children)
        ans.push_back(p.first);
    sort(ans.begin(), ans.end());
    return ans;
}

void mkdir(string path) {
    vector<string> strs = getStrs(path);
    TrieNode *curr = root;
    for (string &str : strs) {
        if (!curr->children.count(str))
            curr->children[str] = new TrieNode();
        curr = curr->children[str];
    }
}

void addContentToFile(string filePath, string content) {
    vector<string> strs = getStrs(filePath);
    TrieNode *curr = root;
    for (string &str : strs) {
        if (!curr->children.count(str))
            curr->children[str] = new TrieNode();
        curr = curr->children[str];
    }
```

```
        curr->isFile = true;
        curr->content += content;
    }

    string readContentFromFile(string filePath) {
        vector<string> strs = getStrs(filePath);
        TrieNode *curr = root;
        for (string &str : strs)
            curr = curr->children[str];
        return curr->content;
    }
};
```