588. Design In-Memory File System

Hard 797 95 Add to List Share

Design a data structure that simulates an in-memory file system.

Implement the FileSystem class:

- `FileSystem()` Initializes the object of the system.
- `List<String> ls(String path)`
  - If `path` is a file path, returns a list that only contains this file's name.
  - If `path` is a directory path, returns the list of file and directory names **in this directory**.

  The answer should in **lexicographic order**.
- `void mkdir(String path)` Makes a new directory according to the given `path`. The given directory path does not exist. If the middle directories in the path do not exist, you should create them as well.
- `void addContentToFile(String filePath, String content)`
  - If `filePath` does not exist, creates that file containing given `content`.
  - If `filePath` already exists, appends the given `content` to original content.
- `String readContentFromFile(String filePath)` Returns the content in the file at `filePath`.

Example 1:

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.

```
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
FileSystem fileSystem = new FileSystem();
fileSystem.ls("/");                         // return []
fileSystem.mkdir("/a/b/c");
fileSystem.addContentToFile("/a/b/c/d", "hello");
fileSystem.ls("/");                         // return ["a"]
fileSystem.readContentFromFile("/a/b/c/d"); // return "hello"
```

Constraints:

- `1 <= path.length, filePath.length <= 100`
- `path` and `filePath` are absolute paths which begin with `'/'` and do not end with `'/'` except that the path is just `"/"`.
- You can assume that all directory names and file names only contain lowercase letters, and the same names will not exist in the same directory.
- 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.
- `1 <= content.length <= 50`
- At most `300` calls will be made to `ls`, `mkdir`, `addContentToFile`, and `readContentFromFile`.

Accepted 45,991 | Submissions 96,806

Seen this question in a real interview before? Yes No

Companies

0 ~ 6 months 6 months ~ 1 year 1 year ~ 2 years

Amazon | 11 Microsoft | 8 Google | 5 Tesla | 3 Airbnb | 2 Salesforce | 2

Related Topics

Hash Table String Design Trie

Similar Questions

LRU Cache Medium

LFU Cache Hard

Design Log Storage System Medium

Java Autocomplete

```java
class FileSystem {

    public FileSystem() {

    }

    public List<String> ls(String path) {

    }

    public void mkdir(String path) {

    }

    public void addContentToFile(String filePath, String content) {

    }

    public String readContentFromFile(String filePath) {

    }
}

/**
 * Your FileSystem object will be instantiated and called as such:
 * FileSystem obj = new FileSystem();
 * List<String> param_1 = obj.ls(path);
 * obj.mkdir(path);
 * obj.addContentToFile(filePath,content);
 * String param_4 = obj.readContentFromFile(filePath);
 */
```