Assignment 8: Directory Trees

CS 301

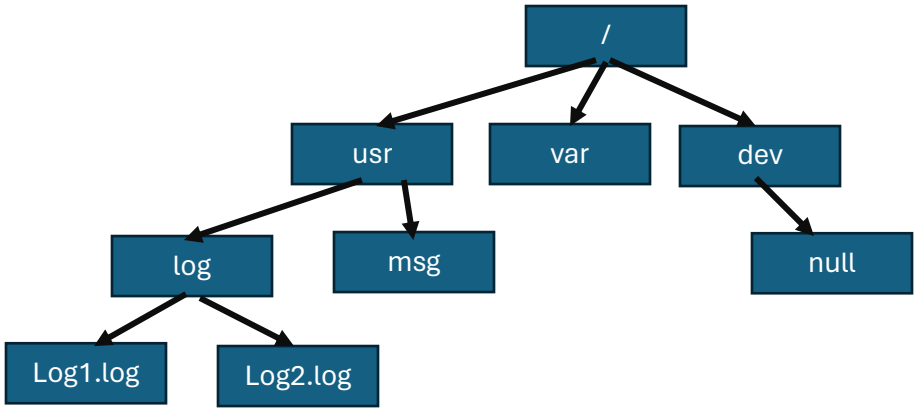
March 6, 2024

Given that you have now gotten some information about the tree data structure, lets try implementing one. More specifically, we will be developing 'directory' trees, which let you represent filesystems and directory structures. It will be helpful for you to look at Chapter 6 from Sections 1-7 in the textbook to get a general idea of how to code your directory tree. But note a key difference from the binary tree code you see in the textbook, in that a directory tree node does not have a limit on the number of children it can have.

Implement a directory tree class called `DirectoryTree` in a Python file named `DirectoryTree.py` with the following methods. Your class may be defined in a recursive fashion or may have helper classes for individual nodes. Regardless, each node must store a string value called `data,` along with references to all of its children, if any:

- `__init__(root='/')` creates a new DirectoryTree, with a root node designated the default string value `'/'.`
- `insert_file(path)` inserts a file into the DirectoryTree following the path defined in the string value `path`. For instance, if `path='/usr/log1.log',` then the function inserts the file by:
    - Using the string to walk through all directories in the path as they appear, if a directory is not found, it is created and inserted into the directory tree. So for a new DirectoryTree with only `'/'`, the method will create the `'usr'` directory, add it as a child of `'/'`, then move forward.
    - Creating a node containing the file `'log1.log'` and insert it as a child of `'usr'.`
- `delete_file(path)` deletes a file/directory in a DirectoryTree following the `path.` For instance, if `path='/usr/log1.log'.` then the method will walk through each directory till it gets to the last, and then delete the file if it exists. If any directory in the path does not exist, or the file does not exist, the method returns an `IndexError` with the message: `'File Not Found'`

- `move_file(source_path,destination_path)` moves the directory/file at end of `source_path`, to the directory path indicated at `destination_path` if it exists. If the destination or source path does not exist, the method should return an `IndexError` with the message: `'Source/ Destination does not exist'`. The directory/file should no longer exist at `source_path`.
- `print_tree()` prints the directory tree structure using the following rules:
    - each directory/file name is on a new line
    - the root of the directory is the leftmost name in the final output, and each subdirectory is shifted one tab space to the right based on its level. As an example, the directory tree given below should be printed as:



OUTPUT:

```
/
	usr
		log
			Log1.log
			Log2.log
		msg
	var
	dev
		null
```

To help you develop your directory tree class: use the following file paths to test out your tree creation.

```
/usr
/var/log/log1.log
/var/log/log2.log
/dev/null
/dev/code/include
/dev/code/include/lib
/usr/patch
/usr/patch/log/log2.log
/usr/patch/log/log3.log
/mnt/c/users/local/program_data/VSCode/data.py
```

Resulting output from print_tree() for this example tree:

```
/
        usr
                patch
                        log
                                log2.log
                                log3.log
        var
                log
                        log1.log
                        log2.log
        dev
                null
                code
                        include
                                        lib
        mnt
                c
                        users
                                local
                                        program_data
                                                        VSCode
                                                                data.py
```