Maciej Kowalczyk

HW4

Report

Functionality 1 & 2

So going into this assignment, I started coding with recursion in mind right away. I began by setting the parameters for the function to be the directory name and 4 variable pointers that will store the sizes and names of the files respectively. The reason is because in the instance that we find a new subdirectory with larger files, we can still keep track of all 4 of the things we have to return without using any helper functions. We take the directory and concatenate the file onto the file path, that way we can check the size and the modes of the file we are currently looking at. Our first comparison checks whether it's a regular file or a directory, if it's a regular file, we then check whether or not that file is writable. After, we check if the file has a larger size, if so, we update our variables that we passed to the function. Going back to first comparison, if its a directory, we do a recursive call with the current filepath (remember us updating the input directory to be a full filepath to whatever we are looking at), and the variables we passed through the first time to keep track of what the largest files are.

Functionality 3 & 4

This was easy to implement, considering our code structure thus far. To begin, we extend the function to take one more parameter which will keep track of the total disk usage. We already know how to find the size of all files within a directory, so every while loop iteration, we have to add the size of whatever we are looking at to the variable we pass into the function. This will return an incorrect size, because we have to consider the size of the root directory as well. Since all directories are of size 4096, we have to initialize our tracker variable to that value and then we get our correct result. The recursive implementation is already there, we just have to pass that pointer to our recursive call.