Andy Ninh

CSC 3350

Lab 3

a. To use the program, you would begin by doing the make command which will create the executable file. After creating the executable file, you would run it by doing the command: .**/file_search <search term> <starting directory>** which will allow you to find the search term of your choice in a selected directory. The parent in the program will start at the current directory location and then find the selected file in given location, going through the different directory paths as needed. Each thread in the program will only go through a quarter of the given length of the provided array, due to it having only 4 threads total. The reason why this is like this would be for each thread searching through the same amount of information in the provided array.

b. The performance of the multi-threaded version of the program is one that takes a longer amount of time, compared to the original version. This is due to having to call directory paths and files recursively, going through each of the different threads.

i. The multi-threaded version doesn't show a 4x speedup because, it takes a recursive approach of searching through the directory and files, with each item being put into an array. Each thread will then revisit each array with all 4 threads and read the calculated divided index. In general, multi-threading with N threads does not automatically lead to Nx speedup because, of having to take additional time to set up the threading, to create the threads, along with the recursive time.

ii. The performance of the multi-threaded version does depend on the directory structure that is being searched. This is due to the directory structure being iterated recursively, with the threaded implementation depending on the directory structure.

iii. The performance of the multi-threaded version could be improved if I managed the thread creation differently, because of how the threads are used. They are used globally when printing the array, instead of visiting paths during the recursion process. Threads could be used as part of the global array in a queue where the threads would add items to the queue as necessary.

c. Running the same program twice in quick succession (usually results in better performance the second time, because of how the operating system stores some information from previous runs in the cache, allowing for a faster file access in the current and future runs over prior ones.