Joshua McMorris

6/5/18

CSC 3350

Dingler

Homework 5/Lab 4b

a. i. There is large amount of additional work required to create the threads and handle all of the backend work required to keep track of and actually use the threads. Additionally, due to the poor optimization of my solution, specifically on files which are not directories (which will run quickly) there will be significant waste in constructing a thread and loss of time before the thread is reallocated. If the program is heavily CPU-bound and long, threading will approach a true 4x speedup but that is unrealistic in reality. There is a lot of work required to make threading work and a lot of things have to be right in order to even achieve performance improvement, not to mention full 4x improvement.
   ii. On my solution performance of the threaded solution is significantly dependent on the starting directory. If there is one directory with 10,000 inner files and only single files in the rest, one threaded will search through the big directory and the rest of the threads will be wasted. Furthermore, this program is VERY poor with files rather than searching directories. It creates a thread and wastes a lot of time just to search one file name against a string.
   iii. I am absolutely not doing this very efficiently. A couple ways performance could be improved is only running a thread on directories (not on other file types) or using a signal system or some other solution to not waste the threads time in the loop until tryjoin is run.

b. I think the optimal workload for my solution would be something like four directories each with a huge number of inner files/directories. This way, only four threads would be created (no wasted time) and the threads would actually operate with the recursive function like the normal recursive solution. Another example of a workload is one in which there are significantly more populated directories than there are normal files in the starting directory. This would mean the threads actually run for a more significant amount of time.

c. Concurrent function runs may perform better as the files' pages being searched for are more likely to already be in memory. This would avoided a needed page read and load from disk (which is super slow). As in chapter 41.4 from the book "The basic mantra is simple: *keep related stuff together*" (page 5). However the server's operating system impelments files, it is likely related files are stored in the same few pages. This would further ensure they would be in pages already loaded in memory.