Nikhil Solanki

CS 361

Homework 7

ns3335@drexel.edu

Instructions to Run Code:

1. Run `make` to build the executable. This will create an executable named `fileSearch` in the current directory. To start a search, use the following command:

```
./fileSearch "target_string" [path_to_directory]
```

`"target_string"`: The text string you want to search for within the files. `[path_to_directory]`: The path to the directory where the search should be performed. If omitted, the current directory will be used.

Question 1

For the File Search tool, I utilized 2 data structures. I first used a `vector` (`std::vector`) to store the paths of the files to be processed, from my `locateFiles()`. Similar to Professor Boady's code and lecture material, I used a `queue` (`std::queue`) for managing the files in a First-In-First-Out order as the `worker` threads process them (`worker()`).

Question 2

I followed Professor Boady's advice in office hours about having a singular thread (`T_0`; `producer()`) locate the files in the directory and push them to a queue. From there, each `worker()` thread from the thread pool then retrieves file paths from this queue and processes them independently, ensuring that their work is done concurrently.

Question 3

The thread pools knows when it is done with all tasks when the file queue is empty, and the producer threads (`producer()`) has finished adding the new paths to the queue. I used a boolean flag (`allFilesListed`) and a condition variable (`cvQueue`), which the worker threads check. When both conditions are true (queue is empty and no more files being added), each worker thread terminates.

Question 4

I used locks in a few locations to ensure thread safety when accessing and modifying the shared file queue. Specifically, a mutex is employed to protect the shared queue. For example, I used a lock in the `worker()` to access and modify the queue safely and access the files in the queue in `producer()`. I also used a lock for printing when the thread found the target string in one of the files in the directory. This is because I was getting sporadic print statements and Professor Boady suggested that we use a lock for printing.

Question 5

I used condition variables to manage the state of `worker()` threads efficiently. Similar to Professor Boady's code (`taskQueue.h`), I used a condition variable to put worker threads to sleep when the queue is empty and wake them up as soon as new files are available in the queue/when all files have been pushed to the queue (`worker()` & `producer()`).