Jason Valenzuela

CS 4414 Project 2 Write-Up

Overview

My project was set up in two stages: parsing and threading. This allowed for discrete parsing of the input from the user and checking for errors before the threads were created.

The parsing stages checked that there were the correct number of arguments input into the program. It then checked that the file contained characters, and was not empty. The file was then checked for any invalid characters, particularly any symbols or alphabetic characters that cannot be converted to an integer. Once the checks were cleared, the first line of the file is read and parsed by spaces. This parsed information is stored in a integer array and passed to the threading stage

The threading stage stores information in two different structs: the thread struct and argument struct. The threading struct contains all the information necessary for each thread to make the computations to compare integers (pthread id, array indices, final value returned). The argument struct contains all the information needed for the routine to function (i.e. number of threads and rounds, binary semaphores, etc.). This is passed to the threading function as a void pointer and then cast as a argument struct.

Implementing the Barrier

The barrier was implemented with 3 binary semaphores, with one mutex and two which acted as turnstiles. The semaphores were initialized before the threads required for the program to run was created.

The barrier was started before and during the max integer comparisons within the array. This was done to ensure that all comparisons were finished before the final computations were finished. In the first barriers, while the semaphore count (integer called ‘scount’) equaled the number of threads, the second turnstile was locked and the first one was unlocked. The second barrier was implemented in the same way, but the second turnstile was unlocked and the first one was locked.

The final part of the function compared the integers in rounds. The rounds were originally calculated with logarithm base 2, but this created some undesired outputs. Dividing the numbers of threads by 2 resulted in a more accurate number of rounds required to implement this function. Each thread had an assigned index value where it would perform the comparision with another integer in the array. Once the greatest number was found, it would write the integer to another final array to the index half of its assigned index (for example, an assigned index 2 would write to index 1 in the final array). Once these comparisons were finished, the main thread would compare within the final array to calculate the maximum number in the final array, and that was returned.

Testing the Barrier

The barrier was tested with several printf() statements at critical points during the computation. It tested to make sure that the thread was successfully created, entered the barrier, did the intended computation, exited the barrier, and exited the program successfully.