Eric Gonzalez

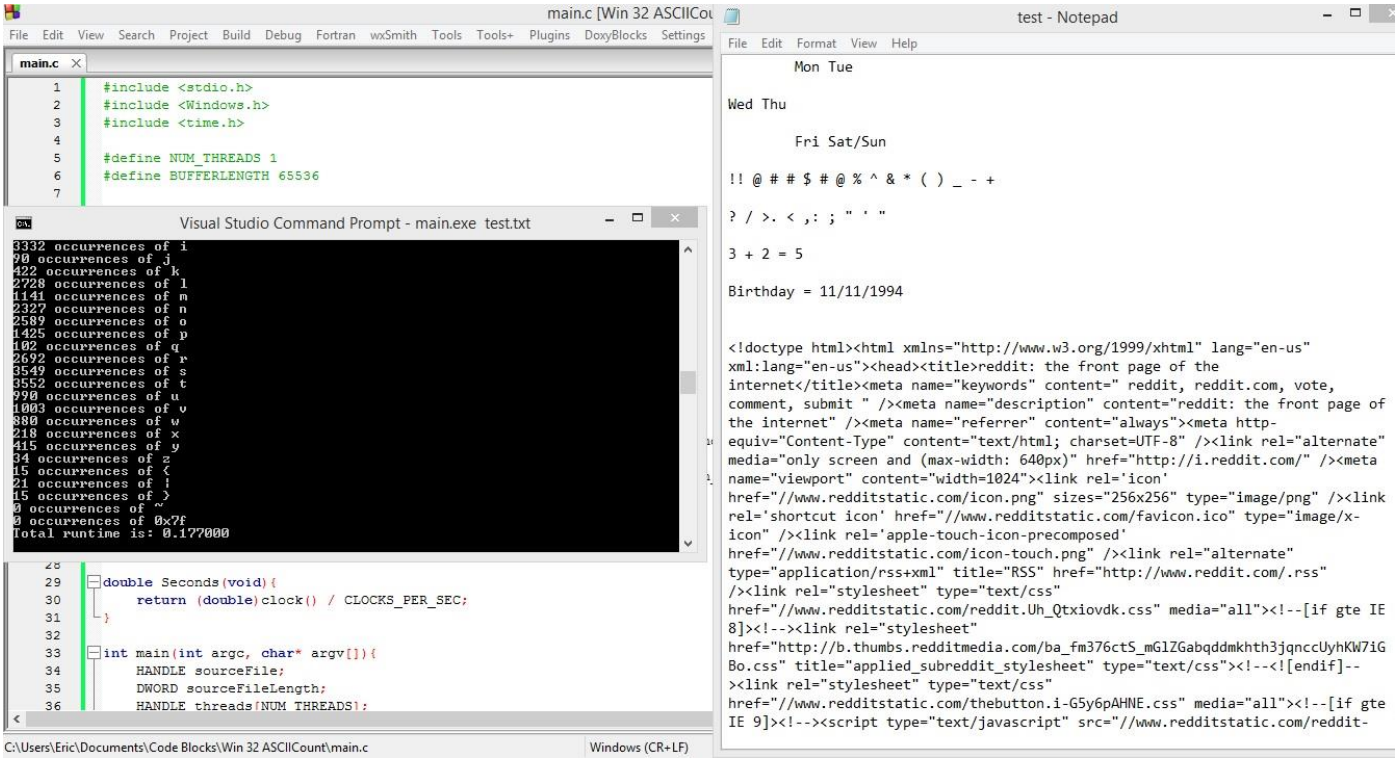
CSC 415

Homework #4

For this assignment we were instructed to implement a multi-threaded ASCII character count program. Basically the program will receive the name of an ASCII text file as a command-line argument and output the number of each kind of ASCII character in the file. I set up a 64KB buffer in global memory and read the contents of the file into the buffer or as much of it as will fit. Then I partition the buffer and spawn a number of threads to count the number of occurrences of each ASCII character in each partition. Each thread recorded these counts in a separate 128-element int array in global memory. When the worker threads finish writing their local counts, they exit. The main thread waits for all of them to complete. It then adds the partition counts for each ASCII character and prints each overall count. I have provided results below.

Win 32

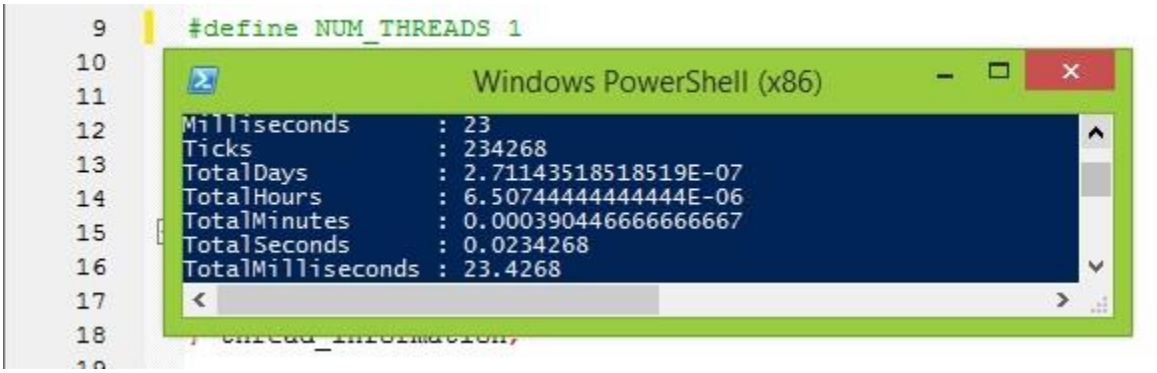
-First I compiled my program and then executed it with a test.txt file. The test file has random input as well as website source code to see a difference in time when testing different number of threads. You can see the test file and output of program in the screenshot below.



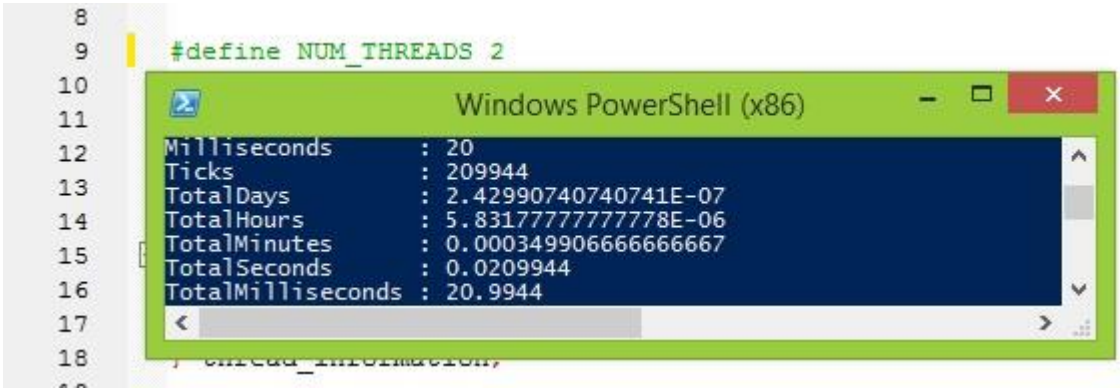
Results of testing different number of threads on a 2 core platform

(I used Measure-Command in PowerShell to measure the time)

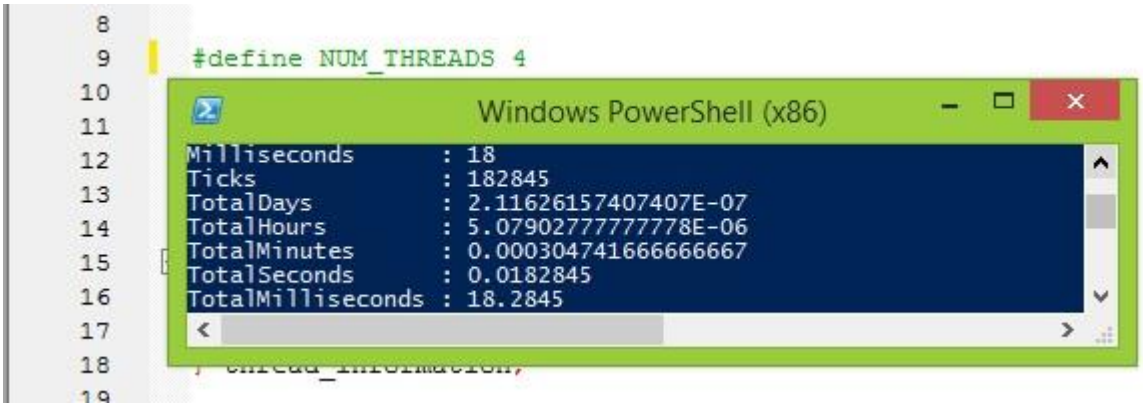
1 thread = 0.023 sec



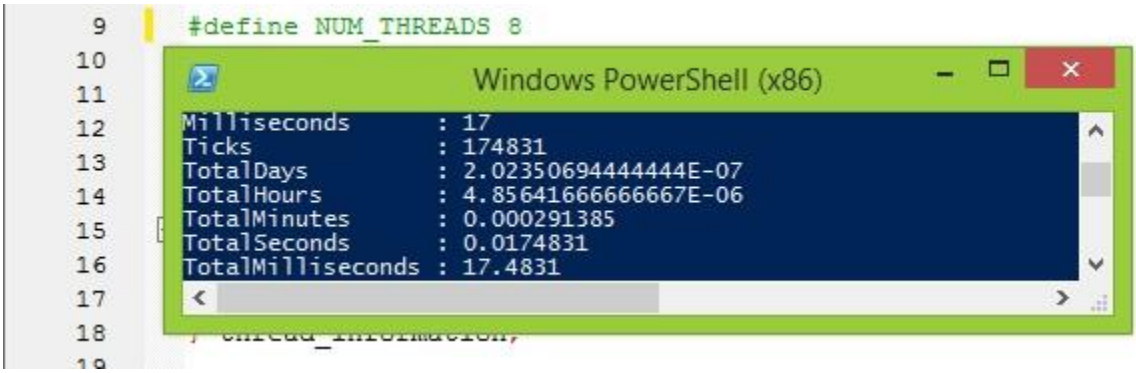
2 threads = 0.020 sec



4 threads = 0.018 sec

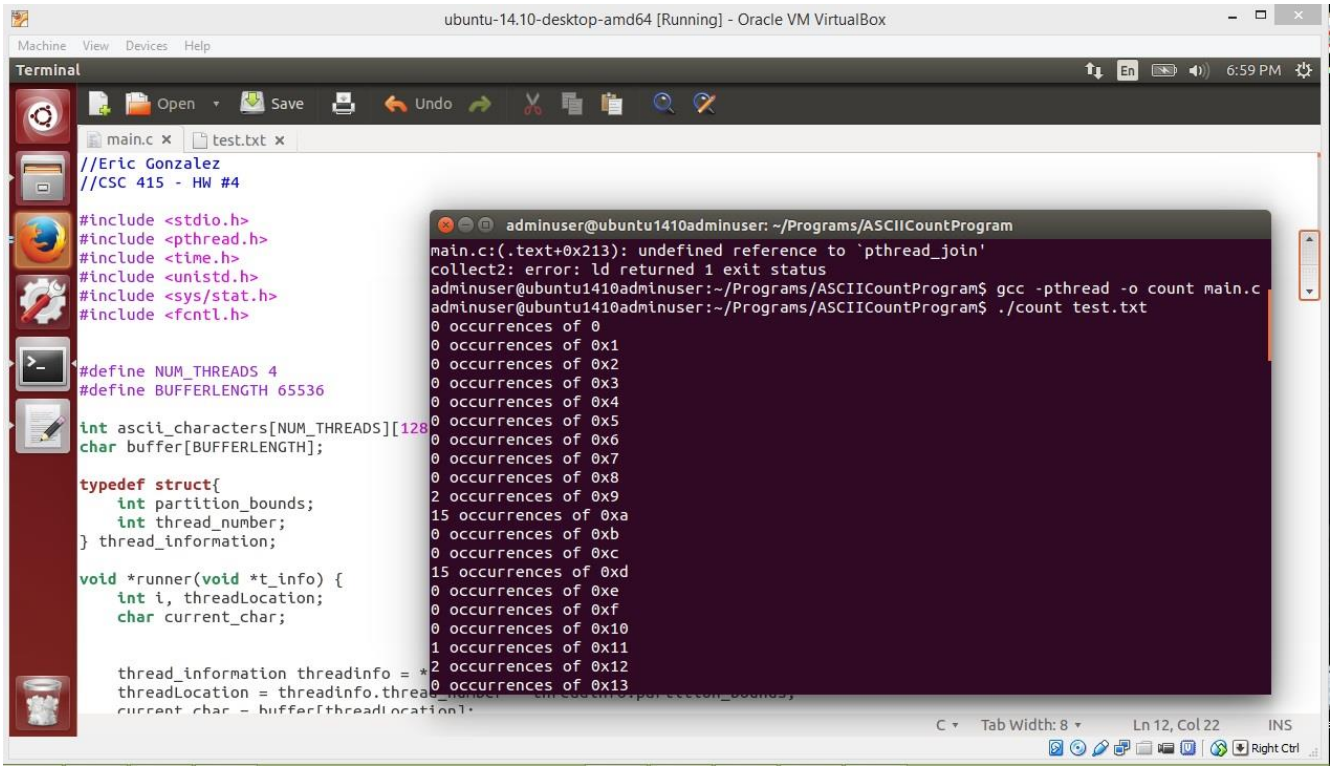


8 threads = 0.017 sec



Posix

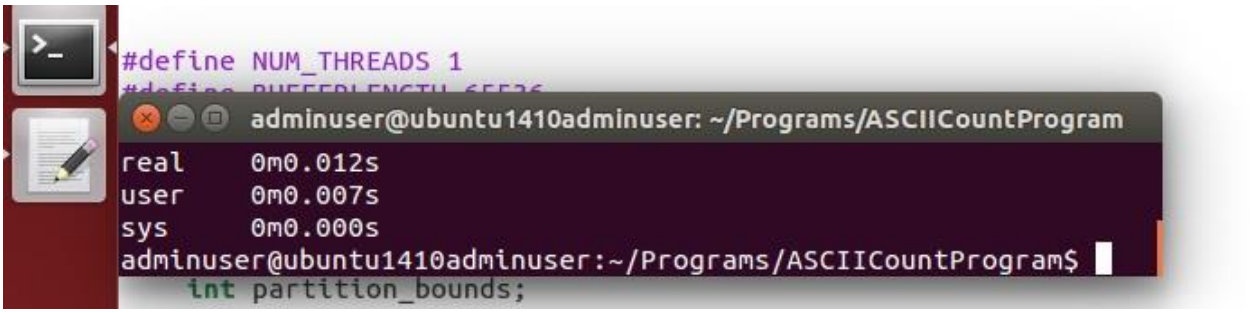
-First I compiled my program and then executed it with a test.txt file. The test file is the same one I use for the win 32 version and has random input as well as website source code to see a difference in time when testing different number of threads. You can see the test file and output of program in the screenshot below.



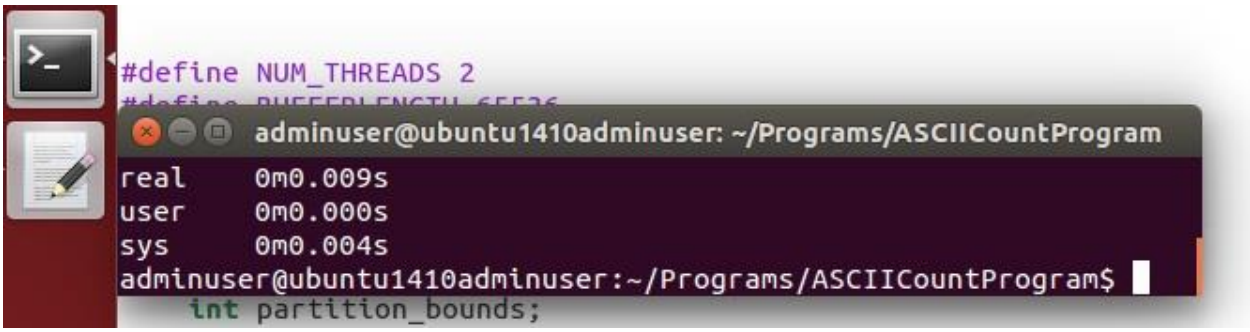
Results of testing different number of threads on a 2 core platform

(I used the Time command to measure the time)

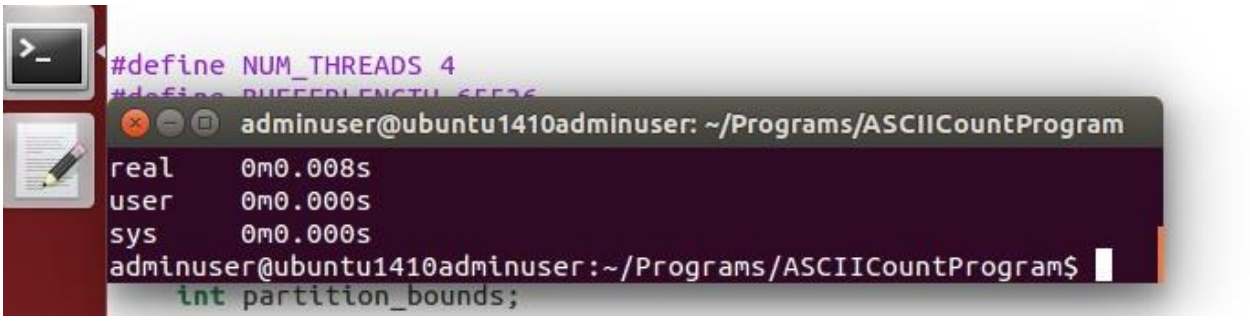
1 thread = 0.012 sec



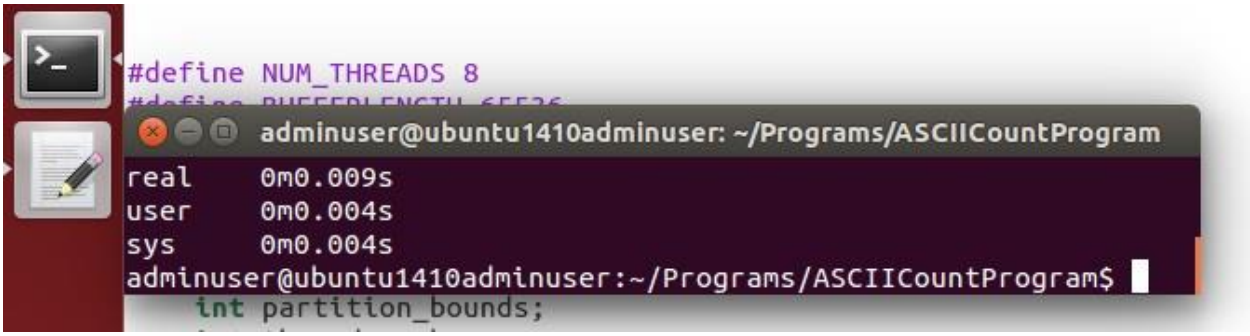
2 threads = 0.009sec



4 threads = 0.008 sec



8 threads = .009 sec



As you can see the execution times get a bit shorter when more threads are used. In some cases you can’t even tell the difference. The speedup is there when more threads are used but it’s really not that noticeable unless using an extremely large file to scan. To some degree you could say it’s linearly improving.