Kendrick Kwok
CSC 415 912351666
//********Assignment 4 :: ASCII Character Count 2.0 Due: 4:00 PM, Thursday, 10/20/2016**

This program taught us to implement a multi-threaded ASCII character count program and compare the results with and without synchronization.

I updated the code for Assignment 3. I changed the arrays storing the counts to a one dimensional array. This is to create code without synchronization. I then created a new file and created a mutex. This mutex lock protects the increment of the global int array using a mutex lock. I ran both versions using the same file and compared the results.

There are two files. One contains a newly created one dimensional array without synchronization, and the other is synchronization with a mutex lock. Mutex lock has a larger amount of counts. Why does this happen and what is the difference between the counts produced with and without synchronization?

The main reason for using a mutex lock is to provide mutual exclusion. This means that when two or more processes are running at the same time, they are attempting to access the same array at the same time. We want the increment of counts to be an atomic operation. An atomic operation is when there is a guarantee that the hardware cannot be interrupted once the code has been executed. There are eight threads that are being executed in this program. When there are more than two threads calling the function count, synchronization problems arises when many threads attempt to write into the array at the same time. Hence, the atomicity is broken, as threads can interrupt each other or work uncoordinated at the wrong times. This is why a mutex lock is so helpful in that once a single thread has reached it's critical section, a mutex lock is called to lock the other threads from executing the function. Until the thread has finished their critical section, which is incrementing the counts of the array in this case, the thread unlocks the lock and allows another thread to enter. This is repeated until all threads have finished and then the mutex lock is destroyed. This way, the final results are accurate.

We can see from the two outputs with and without synchronization, that the one with the mutex lock and has the larger amount of counts. The occurrences are bolded.

Take the highest frequency in the file “a” into consideration. In the program without synchronization, “a” is counted 30211 times, while the program with synchronization with the mutex lock is counted 31312 times. There is a 1000 count difference. You can see that the program with mutex lock has more occurences, and this is because the function count is not being overwritten by multiple threads creating synchronization problems. With the mutex lock, the threads are able to correctly increment the count in the array.

//********OUTPUT**

//********EXECUTION OF MULTITHREAD PROGRAM WITHOUT SYNCHRONIZTION**

kendrick@kendrick-Latitude-D620:~/Desktop$ gcc -pthread -std=c99 ASCII2.c
kendrick@kendrick-Latitude-D620:~/Desktop$ ./a.out pok.txt

1 occurences of 0*0	0 occurences of ,	0 occurences of Y
0 occurences of 0*1	0 occurences of -	0 occurences of Z
0 occurences of 0*2	0 occurences of .	0 occurences of [
0 occurences of 0*3	0 occurences of /	0 occurences of \
0 occurences of 0*4	0 occurences of 0	0 occurences of]
0 occurences of 0*5	0 occurences of 1	0 occurences of ^
0 occurences of 0*6	0 occurences of 2	0 occurences of _
0 occurences of 0*7	0 occurences of 3	0 occurences of `
0 occurences of 0*8	0 occurences of 4	**30211 occurences of a**
0 occurences of 0*9	0 occurences of 5	**16483 occurences of b**
474 occurences of 0*10	0 occurences of 6	**7917 occurences of c**
0 occurences of 0*11	0 occurences of 7	**5264 occurences of d**
0 occurences of 0*12	0 occurences of 8	**1333 occurences of e**
0 occurences of 0*13	0 occurences of 9	**1581 occurences of f**
0 occurences of 0*14	0 occurences of :	0 occurences of g
0 occurences of 0*15	0 occurences of ;	0 occurences of h
0 occurences of 0*16	0 occurences of <	0 occurences of i
0 occurences of 0*17	0 occurences of =	0 occurences of j
0 occurences of 0*18	0 occurences of >	0 occurences of k
0 occurences of 0*19	0 occurences of ?	0 occurences of l
0 occurences of 0*20	0 occurences of @	0 occurences of m
0 occurences of 0*21	0 occurences of A	0 occurences of n
0 occurences of 0*22	0 occurences of B	0 occurences of o
0 occurences of 0*23	0 occurences of C	0 occurences of p
0 occurences of 0*24	0 occurences of D	0 occurences of q
0 occurences of 0*25	0 occurences of E	0 occurences of r
0 occurences of 0*26	0 occurences of F	0 occurences of s
0 occurences of 0*27	0 occurences of G	0 occurences of t
0 occurences of 0*28	0 occurences of H	0 occurences of u
0 occurences of 0*29	0 occurences of I	0 occurences of v
0 occurences of 0*30	0 occurences of J	0 occurences of w
0 occurences of 0*31	0 occurences of K	0 occurences of x
0 occurences of 0*32	0 occurences of L	0 occurences of y
0 occurences of	0 occurences of M	0 occurences of z
0 occurences of !	0 occurences of N	0 occurences of {
0 occurences of "	0 occurences of O	0 occurences of |
0 occurences of #	0 occurences of P	0 occurences of }
0 occurences of $	0 occurences of Q	0 occurences of ~
0 occurences of %	0 occurences of R	
0 occurences of &	0 occurences of S	
0 occurences of '	0 occurences of T	
0 occurences of (	0 occurences of U	
0 occurences of)	0 occurences of V	
0 occurences of *	0 occurences of W	
0 occurences of +	0 occurences of X	

//********OUTPUT**

//********EXECUTION OF SAME ASCII COUNT PROGRAM WITH MUTEX LOCK**

kendrick@kendrick-Latitude-D620:~/Desktop$ gcc -pthread -std=c99 ASCIIMutex.c
kendrick@kendrick-Latitude-D620:~/Desktop$ ./a.out pok.txt

1 occurences of 0*0	0 occurences of ,	0 occurences of Y
0 occurences of 0*1	0 occurences of -	0 occurences of Z
0 occurences of 0*2	0 occurences of .	0 occurences of [
0 occurences of 0*3	0 occurences of /	0 occurences of \
0 occurences of 0*4	0 occurences of 0	0 occurences of]
0 occurences of 0*5	0 occurences of 1	0 occurences of ^
0 occurences of 0*6	0 occurences of 2	0 occurences of _
0 occurences of 0*7	0 occurences of 3	0 occurences of `
0 occurences of 0*8	0 occurences of 4	**31312 occurences of a**
0 occurences of 0*9	0 occurences of 5	**17500 occurences of b**
475 occurences of 0*10	0 occurences of 6	**8014 occurences of c**
0 occurences of 0*11	0 occurences of 7	**5320 occurences of d**
0 occurences of 0*12	0 occurences of 8	**1333 occurences of e**
0 occurences of 0*13	0 occurences of 9	**1581 occurences of f**
0 occurences of 0*14	0 occurences of :	0 occurences of g
0 occurences of 0*15	0 occurences of ;	0 occurences of h
0 occurences of 0*16	0 occurences of <	0 occurences of i
0 occurences of 0*17	0 occurences of =	0 occurences of j
0 occurences of 0*18	0 occurences of >	0 occurences of k
0 occurences of 0*19	0 occurences of ?	0 occurences of l
0 occurences of 0*20	0 occurences of @	0 occurences of m
0 occurences of 0*21	0 occurences of A	0 occurences of n
0 occurences of 0*22	0 occurences of B	0 occurences of o
0 occurences of 0*23	0 occurences of C	0 occurences of p
0 occurences of 0*24	0 occurences of D	0 occurences of q
0 occurences of 0*25	0 occurences of E	0 occurences of r
0 occurences of 0*26	0 occurences of F	0 occurences of s
0 occurences of 0*27	0 occurences of G	0 occurences of t
0 occurences of 0*28	0 occurences of H	0 occurences of u
0 occurences of 0*29	0 occurences of I	0 occurences of v
0 occurences of 0*30	0 occurences of J	0 occurences of w
0 occurences of 0*31	0 occurences of K	0 occurences of x
0 occurences of 0*32	0 occurences of L	0 occurences of y
0 occurences of	0 occurences of M	0 occurences of z
0 occurences of !	0 occurences of N	0 occurences of {
0 occurences of "	0 occurences of O	0 occurences of |
0 occurences of #	0 occurences of P	0 occurences of }
0 occurences of $	0 occurences of Q	0 occurences of ~
0 occurences of %	0 occurences of R	
0 occurences of &	0 occurences of S	
0 occurences of '	0 occurences of T	
0 occurences of (	0 occurences of U	
0 occurences of)	0 occurences of V	
0 occurences of *	0 occurences of W	
0 occurences of +	0 occurences of X	