Homework 4 - ASCII Character Count

Due by 7:00 p.m. Wednesday, 4/1/15

For this assignment, you will implement a multi-threaded ASCII character count program. Your 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. In your code, set up a 64KB (65,536 bytes) buffer in global memory and read the contents of the file into the buffer (or as much of it as will fit). Then partition the buffer and spawn a number of threads to count the number of occurrences of each ASCII character in each partition. (Use a `define` preprocessor directive to specify the number of threads.) Have each thread record these counts in a separate 128-element `int` array in global memory (stored collectively as a 2-dimensional `int` array). The partition bounds can be calculated in the main thread to be roughly equal in size and then passed to each worker thread using a `struct`. You will also need to pass a thread index to each thread so that it knows where to write its local count in the global 2-dimensional `int` array. Once the worker threads finish writing their local counts, they should exit. After spawning the worker threads, the main thread should wait for them all to complete. It should then add the partition counts for each ASCII character and print each overall count. (For non-printable and white space ASCII characters, you should just print the hexadecimal character code.) Here is a sample execution:

```
~$ ./ASCIICount text_file
0 occurrences of 0x0
0 occurrences of 0x1
...
1032 occurrences of 0x20
5 occurrences of '!'
...
```

The POSIX version of your program should use the `pthread_attr_init()`, `pthread_create()`, `pthread_exit()` and `pthread_join()` functions; follow Figure 4.9 in the book. The Win32 version should use `CreateThread()`, `WaitForSingleObject()` and `CloseHandle()` functions; follow Figure 4.10. Test each version of your program using 1, 2, 4 and 8 threads, and record the time taken by each using the `time` command for the POSIX version or the `Measure-Command` command (available in the PowerShell) for the Win32 version. Also report the number of cores on each test platform.

You should submit your source code files (one for each platform) and a short writeup in pdf format that includes a description of what you did and the compilation and execution output from each of your programs. Also discuss the execution times as a function of the number of threads and number of cores on each platform. (What is the speedup observed? Is it linear?) Submit everything to the regular submission link on iLearn, and then submit just the writeup to the TurnItIn link to generate an originality report.