CSC415 OPERATING SYSTEM PRINCIPLES

Assignment 3
ASCII Character Count

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 `#define` directive to set the number of threads to 8.) 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, i.e., ASCII characters with decimal values smaller than 33, you should just print the hexadecimal character code.)

Name your C file as “ASCIICount”. When you compile it, you need to add `-pthread` to the command

```
gcc ASCIICount.c -pthread
```

Then, use the following the command to run the code with test file

```
./a.out test.txt
```

Here is the execution result with the test file:

```
0 occurrences of 0x0
0 occurrences of 0x1
...
0 occurrences of '!'
...
500 occurrences of 'A'
500 occurrences of 'B'
```

```
500 occurrences of 'C'
500 occurrences of 'D'

...
```

Hints:

1. You can use the function `fopen()` to open a file, and then use `fread()` to read the contents of the file into a buffer. You can find the description and examples of `fopen()` and `fread()` here http://www.cprogramming.com/tutorial/cfileio.html

2. You should use the functions `pthread_attr_init()`, `pthread_create()`, `pthread_exit()` and `pthread_join()` to create and manage the threads. I also attach the example (i.e., Fig. 4.9 in the textbook) we discussed in class for your reference.

Submit the source code file to the regular submission link on iLearn.