Jacob Schwell

Professor Foreman,

CS-311

October 6, 2017

MTPC Design Document

**Summary:**

The purpose of this project is to use multiple threads in order to read a file, line-by-line, and write it out to another file. The code is written to be able to be compiled on Windows or Linux and work without any modifications.

Pthread for Windows header files, along with time\_functions.h, are located in “C:\Windows\Temp\pthreads”—the library file pthreadVC2.lib is also in that directory. pthreadVC2.dll is in “C:\” drive; however, it can be anywhere in the PATH since it is a dll and the OS will search the PATH for it. The input file is “C:\temp\coursein\p2-in.txt” and the output file is “C:\temp\courseout\p2-out.txt”

Linux has pthreads built in; however, it must be compiled with the -lpthread flag to compile correctly. The file paths for I/O are “/home/student/temp/coursein/p2-in.txt” and “/fileio/p2-out.txt.”

**Int Main(int argc, char\*\* argv):**

The main() function of this program sets up all the requirements for the producer and consumer threads as well as starting and stopping the timer, handling the user-interface, and handling error messages. If the program is unable to open either file, it will print a message and exit. If any errors occur while starting or joining the threads, it will, again, print out an error message and exit the program. Once both threads complete and join back at the main thread, the timer stops. After that it closes out the input and output file streams and displays the CPU and Wall time.

**Void\* readFrom(void\* fromFile):**

The readFrom(void\* fromFile) function is the producer thread. This thread takes in a FILE\* variable type-casted to a void\* to conform to the pthread methods. Once started it loops until it finishes reading the file. During each iteration, it will make sure there is at least one empty buffer slot and that no other thread is in the critical section. Then it reads a line into the current buffer slot index, then posts the critical section semaphore (inUseSem) and the full buffer slot semaphore (fullSem) to keep track of how many slots have new data in them. If the line read is NULL, it did not read anything since it finished reading every line from the file. Once this happens, the thread will set doneReadingFile to true so the consumer will know it is done reading, then it undoes it leaves the critical section and undoes its decrement of the number of empty slots semaphore, since it did not actually read any new data into the buffer.

**Void\* writeTo(void\* toFile):**

The writeTo(void\* toFile) function is the consumer thread. This thread takes in a FILE\* variable type-casted to a void\* to conform to the pthread methods. Once started it loops until the producer finishes reading the file and it finishes writing all the data in the buffer. During each iteration, this thread waits until there is at least one full buffer slot, then enters the critical section once if no other thread is their respective critical section. From there it writes the data to the file, leaves the critical section, and increments the number of empty buffer slots semaphore (emptySem). Before the next iteration, it checks how many full buffer slots there are, so that once the producer finishes reading the file this thread can make sure it finishes writing all the data.