This file discusses Project 3.

Please read all of these instructions as failure to comply may result in a loss of points in your grade.

Your project deliverables are as follows:

1) Only one source code file as follows with filename as in the example: JSvadlenka\_prj3\_sectnn\_src.c (or .cpp)

in which you substitute YOUR OWN NAME for mine. Also replace nn in sectnn with YOUR course section number.

a) At top of the file: your name in comments, for example: // First\_Name Last\_Name

b) Next, the gcc or g++ code compilation and linkage command, for example in my case (not yours) it would be: // gcc -o JSvadlenka\_prj3\_sectnn\_src.c -o JSvadlenka\_prj3\_sectnn\_src.exe

c) In-line code comments no fewer than every 5 lines of code which describe your algorithmic steps. Syntax transcriptions of the code are not considered in-line code comments! (For example, I do not want to see the comment for i++ which says "Increment i.")

2) The source code accepts one command line argument which is the name of the input file to be read. You will use the same two input file from Project 2.

3) Your project write-up of your program design and issues encountered and resolved in a WORD or PDF file in the filename format JSvadlenka\_prj3\_sectnn\_wrt.

4) Your ASCII text results file (filename format JSvadlenka\_prj3\_sectnn\_rslt.txt ) containing output from program execution for each of the two input files.

Project 3 Description

--------------------------

1) Your program will make use of the POSIX system I/O calls and pthreads library discussed in lectures. That is, you must only use open(), close(), pread(), write(), and the pthreads library function calls for multithreading. Your program will have 2 pthreads in a Producer Consumer pattern using a stack for interthread communication. The mainline thread opens the input file to get a file descriptor, and it also creates the 2 pthreads. The main thread waits for each pthread to complete its work by using pthread\_join() and the main thread will subsequently generate all output to screen using write().

The first pthread, the Producer, reads the input file of integers and pushes each integer on to a stack. Meanwhile, the second thread, the Consumer, pops the integers from the stack and keeps track of the minimum integer of all integers popped so far from the stack. After all 10,000 integers have been popped, the minimum integer is obviously the global minimum of the file. You may use any data structure for the stack, but the stack can only hold a maximum of 100 integers at any time. There is no chunking of the input file as in the previous project.

2) Since the stack is a shared data structure between the Producer and Consumer threads, any code that accesses the stack is a critical section and you will use pthread mutex function calls for the entry and exit sections in both the Producer and Consumer threads.

The mutex code you will use is as follows and can also be found in Chapter section 7.3.1 of the textbook.

/\* Create and initialize the mutex lock. \*/

pthread\_mutex\_t myMutex;

pthread\_mutex\_init(&myMutex, NULL);

/\* Entry section: Aquire (lock) the mutex immediatley prior to your thread Critical Section. \*/

if (pthread\_mutex\_lock(&myMutex) != 0) {

/\* Do error handling here as nonzero return code is an error \*/

}

/\* Your critical section of code for the stack is here \*/

/\* Exit section: Release the mutex immediately after your thread Critical Section. \*/

if (pthread\_mutex\_unlock(&myMutex) != 0) {

/\* Do error handling here as nonzero return code is an error \*/

}

/\* Remainder section is here \*/

3) The Consumer Thread will store (to a global integer array of 10 elements) the running current minimum integer so far of all

integers popped after each successive 1,000 integers have been read from the file. This global integer array will be used

later by the main thread to write the output results in item (5).

4) Remember that you will have to use either pthread\_yield or sched\_yield functions as in Project 2 in the Producer and Consumer Threads.

5) As before, you are NOT permitted to use any C++ nor C standard I/O library functions nor cin nor cout. Furthermore, you cannot use

any C++ nor C standard library functions for string operations.

6) OUTPUT: Remember, all thread output goes to the screen (standard out).

a) Main thread writes 10 lines of information ( one line each for every minimum number generated by the Consumer Thread in item (3) above).

Please use the following format for each line:

Output:

Minimum integer after 1000 integers = \t PLACE THE MINIMUM HERE \n

Minimum integer after 2000 integers = \t PLACE THE MINIMUM HERE \n

Minimum integer after 3000 integers = \t PLACE THE MINIMUM HERE \n

Minimum integer after 4000 integers = \t PLACE THE MINIMUM HERE \n

Minimum integer after 5000 integers = \t PLACE THE MINIMUM HERE \n

Minimum integer after 6000 integers = \t PLACE THE MINIMUM HERE \n

Minimum integer after 7000 integers = \t PLACE THE MINIMUM HERE \n

Minimum integer after 8000 integers = \t PLACE THE MINIMUM HERE \n

Minimum integer after 9000 integers = \t PLACE THE MINIMUM HERE \n

Minimum integer after 10000 integers = \t PLACE THE GLOBAL MINIMUM HERE \n