CS39002: Operating Systems Lab
Spring 2012

Assignment 5
Due: March 09, 2012, 1 pm

Write a program that uses POSIX threads to perform a couple operations on Boolean matrices. All shared data should be declared as global variables. Mutual exclusion and thread synchronization must be achieved by pthread library calls only (mutexes and condition variables). You are *not allowed* to use shared memory, semaphores or any other inter-process communication mechanism (like pipes).

Part 1: Generating a Boolean matrix

The master thread creates an $M \times M$ Boolean matrix A with random entries. Each entry of the matrix is zero or one with probability half. The matrix is to be stored in a global two-dimensional array, and may be statically or dynamically allocated (your choice).

Part 2: Creating the worker threads

The master thread creates N worker threads among which the following Boolean matrix operations will be equitably shared. Notice that the same threads will take part in *all* of the following operations. That is, you must not create a fresh set of threads for each individual matrix operation or in each iteration. The master thread must create all necessary resources (like mutexes and condition variables to be used later) *before* the creation of any worker thread. The worker threads must be joinable.

Part 3: Counting the number of ones in A

Each of the N worker threads computes the number of ones in an $(M / N) \times M$ submatrix, and adds this count to a shared (that is, global) counter variable. The master thread initializes the counter to zero before any worker thread accummulates its count in the counter. The updating of the counter by the N threads must be mutually exclusive. After all the worker threads accummulate their respective counts, the master thread prints the value of the counter.

Part 4: Computing the transitive closure of A

For two Boolean matrices U and V, define the product UV as the Boolean matrix W such that the (i,j)-th entry of W is one if and only if there is a k for which the (i,k)-th entry of U and the (k,j)-th entry of V are both one. This multiplication is not the standard modulo-two product of U and V.

The worker threads compute A, A^2, A^4, A^8, . . . , A^{2^e} for some e satisfying $2^e \geq m$. In each squaring step, the current matrix stored in A is multiplied with itself, and the result is temporarily stored in a second global matrix B. Each of the N worker threads computes M / N rows of the product. After all the worker threads complete their respective parts in the computation of B, they collectively copy back the product stored in B to the matrix A. Each worker thread copies the portion of B computed by it back to A.

Since the locations of A and B modified by the worker threads are disjoint from one another, no mutual exclusion is necessary during squaring or copying back. However, the copying of B to A must start *after* all computations involving the old A are over. Moreover, the next squaring step is allowed to start only *after* the entire copy of B to A is over. Use condition variable(s) to synchronize the worker threads.

After A^{2^e} is computed (and stored in the global array A), the master thread prints the matrix A.

Part 5: Winding up

Each worker thread individually terminates at this point. The master thread waits for all the worker threads to join. After that, the master thread cleans up thread resources, and exits.

Submit a single C source file with the name `BoolMat.c`. Take $M = 1000$ (dimension of the matrix A) and $N = 4$ (number of worker threads) in your submission.