# EE 451: Parallel and Distributed Computation

## PA2 — Spring 2021

## Due date: Sunday 7th February 2021 11:59 PM

1. **Examples**

   - A Pthreads example code, "print_msg_with_join.c", which prints a hello message in each thread. To compile and run it,

     ```
     1 gcc -O3 -o run print_msg_with_join.c -lpthread
     2 srun -n1 -c4 ./run
     ```

     Here '-c' specifies the number of CPUs (threads) you will be using. In this case, the code is run with 4 CPUs.

   - A Pthreads example code that uses a mutex, "increase_shared_var.c", which lets 2 threads increase a shared variable. To compile and run it,

     ```
     1 gcc -O3 -o run increase_shared_var.c -lpthread
     2 srun -n1 -c2 ./run
     ```

   - A Pthreads example code that uses a mutex and a condition variable, "increase_shared_var_cond.c", which lets 2 threads *take turns* to increase a shared variable. To compile and run it,

     ```
     1 gcc -O3 -o run increase_shared_var_cond.c -lpthread
     2 srun -n1 -c2 ./run
     ```

2. **Parallel Matrix Multiplication**

   Use Pthreads to implement a parallel version of your naive matrix multiplication code from PA1. Set the number of threads $p = 1, 2, 4, 8$. For each value of $p$, print out the value of $C[100][100]$ and the execution time. Plot the execution time w.r.t. $p$, and briefly discuss your observation (you don't need to plot in the program). Name your program 'problem2a.c'.

3. **Producer-Consumer Problem**

   The producer-consumer problem (also known as the bounded-buffer problem) is a classic example of a multi-thread synchronization problem. In this assignment, you will simulate the producer-consumer problem using a mutex and a condition variable. Producer makes cookies and puts cookies on the shelf. Consumer buys cookies and takes cookies away from the shelf. The simulation works as follows.

   - There are **two** consumers and one producer. Each producer or consumer is implemented as a thread.
   - The capacity of the shelf is 10. Initially, the shelf is empty.

- When the producer checks the number of cookies on the shelf, (1) if the number is less than 9, the producer puts 2 cookies on the shelf; (2) if the number is 9, the producer puts 1 cookie on the shelf.
- When the shelf is not empty, a consumer thread can only take 1 cookie at a time.
- If the producer thread has put $\geq 30$ cookies on the shelf, it can be terminated.
- If a consumer thread has taken 15 cookies away from the shelf, the consumer thread can be terminated.
- Whenever the number of cookies on the shelf changes, you need to print a message in the following format.
  - Producer has put 2 cookies; # of cookies on the shelf changes from 0 to 2.
  - Consumer 2 has taken 1 cookies; # of cookies on the shelf changes from 2 to 1.
  - Producer has put 4 cookies[1]; # of cookies on the shelf changes from 1 to 3.
  - Consumer 1 has taken 1 cookies; # of cookies on the shelf changes from 3 to 2.
  - Consumer 1 has taken 2 cookies[2]; # of cookies on the shelf changes from 2 to 1.
  - Consumer 1 has taken 3 cookies; # of cookies on the shelf changes from 1 to 0.
  - ...

Please implement 2 versions of the simulation:

1. **Using mutex only**
   In this version, when each thread intends to check/update the number of cookies on the shelf, the thread needs to acquire the mutex first in order to avoid any race condition. Here, only mutex is used. Name this program 'problem2b1.c'.
2. **Using mutex and condition variable**
   In this version, when the producer checks the number of cookies on the shelf, if the number is 10, the producer goes to sleep; if the number is 0, the producer puts 2 cookies on the shelf and sends a broadcast signal to wake up the consumers. When a consumer checks the number of cookies on the shelf, if the number is 0, the consumer goes to sleep; if the number is 1, the consumer takes 1 cookie and wakes up the producer. Initially, all threads are awake. Name this program 'problem2b2.c'.

[1]This is an accumulative number. The producer has put 2+2 = 4 cookies so far.
[2]This is an accumulative number. Consumer 1 has taken 1+1=2 cookies so far.

4. **Submission**

- Three .c files: 'problem2a.c' for parallel matrix multiplication; 'problem2b1.c' and 'problem2b2.c' for the producer-consumer problem. Make sure your programs are runnable. (10+10+10 pts)
- Screenshots of the program outputs for parallel matrix multiplication ($p = 1, 2, 4, 8$). (10 pts)
- Plot of execution time and your observation for parallel matrix multiplication. (5 pts)
- Screenshots of the program outputs for the producer-consumer problem (version 1&2). (5 pts)