CS341 #17. Producer Consumer, Semaphores, Condition Variables. Barriers & Reader Writer Problem

1. Producer Consumer & Counting Semaphores (review)

Assume buffer is an array of length 16. Better names for s1?s2?

|  |  |
| --- | --- |
| 1. void add(value) { 2. sem\_wait(&sem\_s1) 3. buffer[ (in++) & 15 ] = value; 4. sem\_post(&sem\_s2); 5. } | 1. remove() { 2. sem\_wait(&sem\_s2); 3. result = buffer[ (out++) & 15 ]; 4. sem\_post(&sem\_s1); 5. return result; 6. } |
| Q. What are 'sem\_s1' and sem\_s2? When do they block?  Q. What should be their initial values?  Q. What if sem\_s1 was only initialized to 7? Would the producer consumer still work? to 32?  Q. What is missing from the above code? When would it matter?  Q. Could you implement a producer consumer queue using condition variables instead? | |

2. Fix the following multithread code to be thread safe, and use condition variables to avoid busy waiting

|  |
| --- |
| 1. #define N (8) 2. pthread\_cond\_t cvs[N]; 3. pthread\_mutex\_t locks[N]; 4. int data[N]; 5. int quit; 6. void init() { 7. for(int i =0; i < N;i++) { 9. pthread\_cond\_init(cvs + i, NULL); 10. pthread\_mutex\_init(locks + i, NULL); 11. } 12. }      1. // Wait until data[i] > 1, then subtract 2 and increment data[i+1] 2. void runner(void\*arg) { // For N-1 threads. Each thread gets a value 0 to N-2 3. int i = (int) i; 4. while(!quit) { 5. while(data[i] < 2) { 6. sleep for a bit 7. } 8. data[i] -= 2; 9. data[i+1] ++; 10. } 11. } 12. void modify(int index, int amount) { 13. data[index] += amount; 14. } |

3.Counting Semaphore Quick Review I. choose {will always / may / will never} :

sem\_post \_\_\_\_\_\_\_\_\_\_\_ block sem\_wait \_\_\_\_\_\_\_\_\_\_\_ block.

3.Counting Semaphore Quick Review II

10 threads call sem\_wait. 3 threads immediately continue, the other 7 are blocked. Then sem\_post is called twice (2). How many additional threads will continue?

4. Three classic / well known synchronization problems:

Barrier

Producer Consumer

Reader-Writer Problem

5. pthread barriers

pthread\_barrier\_init( &barrier, \_\_\_\_\_\_\_);

pthread\_barrier\_destroy(&barrier)

pthread\_barrier\_wait( &barrier)

Return values?

0

PTHREAD\_BARRIER\_SERIAL\_THREAD

|  |
| --- |
| 6. Use a CV to implement a single-use barrier until all 8 threads have reached the barrier. |

7. Post-lecture challenge:

i) Can you make a barrier using only counting semaphores?

ii) Can you make a barrier using only mutex locks?