CS241 Lawrence Angrave L16 – Condition Variables II.

Implementing a barrier. Producer Consumer

1. Condition Variable pop-quiz

Which call do you implement inside a loop?

What is spurious wake up?

Why do you need a mutex too?

How do you wake up one or all blocked threads?

2. Fix the following multithread code. remove should never allow the account to go negative.

pthread\_mutex\_t m;

pthread\_cond\_t cv;

int money = 100;

void init() {

money = 100;

}

void add(int amount) {

money += amount;

}

int remove(int amount)

money -= amount;

return money;

}

3. Three classic / well known synchronization problems:

Barrier

Producer Consumer

Reader-Writer Problem

4. Use a CV to implement a simple version of a *counting* *semaphore*

Note a real semaphore might implement a queue of waiting threads to ensure fairness (and avoid *starvation*).

|  |  |
| --- | --- |
| sem\_init(sem\_t \*s, int shared, int value) {  } | typedef struct sem\_t {  } sem\_t; |
| sem\_post(sem\_t\*s) {  } | sem\_wait(sem\_t\*s) {  } |

5. Use a CV to implement a *barrier* Do not continue to calc #2 until all 16 threads have reached the barrier.

pthread\_mutex\_t m;

­?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

double data[256][8192] ;

int main() {

pthread\_mutex(&m, NULL);

?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

pthread\_t ids[N];

for(int i=0;i<N;i++) pthread\_create( ?\_\_\_\_\_\_\_\_\_\_\_ , NULL , calc, (void\*) i );

// Wait for all threads to finish

for(int i=0;i<N;i++) ?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\* print out result\*/

}

? calc( ­? ) {

/\* Divide matrix work up into blocks of 16 columns.

int x,y, start = 16 \* ?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int end = start + 16;

for(x = start; x<end;x++) for(y=0; y <8192;y++) /\* do calc #1 \*/

// Wait here until all threads have finished calc #1.

for(x = start; x<end;x++) for(y=0; y <8192;y++) /\* do calc #2 \*/

return ?\_\_\_\_\_\_

}