CS241 Lawrence Angrave L17 – Implementing a barrier.  
Reader Writer Problem

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

Why are not just join & create more threads?

pthread\_mutex\_t m;

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

double data[256][8192] ;

int main() {

/\* code to initialize the data values \*/

pthread\_mutex\_create(&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++) ?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

/\* code to print out result\*/

}

Implement the calc function that will be called by 16 threads:

? 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 until all threads have finished calc #1.

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

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

}

Q2 What is the Reader-Writer Problem?

How is it different from the Producer-Consumer Problem?

What is wrong with attempt #1

|  |  |  |
| --- | --- | --- |
| p\_mutex\_t \*readlock=malloc(  p\_m\_init(readlock,NULL)  P\_m\_init(writelock,NULL) | read() {  lock( readlock)  // do read stuff  unlock(readlock)  } | write() {  lock(writelock)  lock(readlock)  // do writing stuff here  unlock(readlock)  unlock(writelock) |

#2 Does this work?

|  |  |  |
| --- | --- | --- |
| int reading=0,writing=0  p\_m\_init(readlock,NULL)  P\_m\_init(writelock,NULL) | read() {  while(writing) {}  reading = true  // do  reading = false  } | write() {  while(reading||writing) {}  writing = true  // do writing stuff here  writing = false  } |

#3 What variables and synchronization primitives do you need for your first implementation?

|  |  |  |
| --- | --- | --- |
| init | read() { | write() { |

Lawrence's first implementation: