CS241 #17. Implementing a barrier. Producer Consumer. Introducing the Reader Writer Problem

1. Producer Consumer (review)

Assume buffer is an array of length 16.

|  |  |
| --- | --- |
| add(value) {  sem\_wait(&sem\_empty)  buffer[ (in++) & 15 ] = value;  sem\_post(&sem\_full); | remove() {  sem\_wait(&sem\_full);  result = buffer[ (out++) & 15 ];  sem\_post(&sem\_empty); |
| Q. What are 'sem\_empty' and sem\_full? When do they block?  Q. What should be their initial values?  Q. What if sem\_full was only initialized to 7? Would the producer consumer still work? 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?  Q. Can you make a queue of work items for threads? | |

2. Fix the following multithread code to thead safe. Additionally, remove should never allow the account to go negative i.e. it should block until there are sufficient funds.

|  |
| --- |
| pthread\_mutex\_t m;  pthread\_cond\_t cv;  int money = 100;  void init() {  money = 100;  }  void add(int amount) {  assert(amount>0);  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 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 ?\_\_\_\_\_\_  } |

5. Fun with the terminal: "VT100 codes"

e.g. ' <ESC>[{ROW};{COLUMN}H' becomes fprintf(stderr, "\033[%d;%dH%s", x, y, mesg)

6. Challenge: Make a barrier using mutex locks

7. Challenge: Make a barrier using counting semaphore