CS241 Lecture 16 Lawrence Angrave   
Semaphores

Condition Variables Review

1. I have two threads blocked on a condition variable 'cv1'

while( cloudy == 42 ) p\_cond\_wait( &cv1, &m);

How do I wake them both up?

p\_m\_lock(&m);

cloudy = 42;

**pthread\_cond\_boardcast(&cv1);**

p\_m\_unlock(&m);

2. What must be locked before calling p\_cond\_wait ?

The mutex!

3. How do I use counting semaphores?

sem\_init(&s, 0, 12);

sem\_post(&s); //increments the count by one

sem\_wait(&s); // may block if no left / reduce count by 1

SAFE IN SIGNAL HANDLER

4. What is a ring buffer?

5. How can I use counting semaphores to implement a ring buffer?

pthread\_mutex\_t m = PTHREAD\_MUTEX\_INITIALIZER:

void init() {

sem\_init( \_&s1\_\_\_\_, 0, \_\_\_\_\_\_);

sem\_init( \_&s2\_\_\_, 0, \_\_\_\_\_\_);

}

void sync\_enqueue(work\_t \*work) {

sem\_wait(&s1)

do enqueue staff

sem\_post(&s2);

}

work\_t\* sync\_dequeue(){

sem\_wait(&s2);

do deque

sem\_post(&s1);

}

Some more C functions for you:

sigprocmask pthread\_sigmask pthread\_self() atexit sigaction

**Psuedo code Candidate** # 1

|  |  |
| --- | --- |
| wait until your flag is lowered  raise my flag  // *Do Critical Section stuff*  lower my flag | wait until your flag is lowered  raise my flag  // *Do Critical Section stuff*  lower my flag |

// Threads do other stuff and then will repeat

Problems with 1?

**Candidate** #2

|  |  |
| --- | --- |
| raise my flag  wait until your flag is lowered  // *Do Critical Section stuff*  lower my flag | raise my flag  wait until your flag is lowered  // *Do Critical Section stuff*  lower my flag |

// Threads do other stuff and then will repeat

Problems with 2?

**Candidate** #3

|  |  |
| --- | --- |
| wait until my turn (turn==id?)  // *Do Critical Section stuff*  turn = *yourid* | wait until my turn (turn==id?)  // *Do Critical Section stuff*  turn = *yourid* |

// Threads do other stuff and then will repeat

Problems with 3?

What have I made?

|  |
| --- |
| 1. pthread\_mutex\_t m = PTHREAD=MUTEX\_INITIALIZER; 2. pthread\_cond\_t cv = PTHREAD\_COND\_INITIALIZER; 3. int cake = 0; 4. void decrement() { // Waits if nonzero 5. lock(&m) 6. while(cake == 0) p\_cond\_wait(&cv, &m); 7. cake --; 8. unlock(&m); 9. } 10. void increment() { 11. lock(&m); 12. cake ++; 13. if( \_\_\_\_\_\_\_\_\_\_\_\_\_ ) p\_cond\_signal(&cv); 14. unlock(&m); 15. } |