**1. Monitor (a lock & Condition variables)로 Semaphore를 구현하시오.**

**Basics**)

P() :an atomic operation that waits for semaphore to become positive, then decrements it by 1

V() : an atomic operation that increments the semaphore by 1, waking up a waiting P, if any

**Semaphore element & function algorithm**)

Int val; //구슬의 개수

Lock lock;

Condition con;

P() {

Lock

Decrement val

If val is negative then unlock & sleep // use condition variable.wait()

unlock

}

V() {

Lock

Increment val

Wake up waiter

Unlock

}

**Building semaphore function**)

P() {

Lock.acquire();

Val--;

If(val<0)

Con.wait(&lock) // unlock&sleep

Lock.release();

}

V() {

Lock.acquire();

Val++;

If(val<=0)

Con.signal();

Lock.relaese();

}

**Flow example**)

Process A, Process B, Process C가 race condition상태에 있다고 할 때

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Val  (init  =1) | Lock  (init  =0) | A | B | C |
| 1  1  0  0 | 0  1  1  0 | P():  Lock.acquire()  Val--;  Lock.release() |  |  |
| 0  0  -1  -1 | 0  1  1  0 |  | P():  Lock.acquire()  Val--;  Con.wait(&lock) |  |
| -1  -1  -2  -2 | 0  1  1  0 |  |  | P():  Lock.acquire()  Val--;  Con.wait(&lock) |
| -2  -2  -1  -1  -1 | 0  1  1  1  0 | V():  Lock.acquire();  Val++;  Con.signal();  Lock.release(); |  |  |
| -1  -1 | 1  0 |  |  | p’() : @con.wait(&lock)  lock.release(); |
| -1  -1  0  0  0 | 0  1  1  1  0 |  |  | V():  Lock.acquire();  Val++;  Con.signal();  Lock.release(); |
| 0  0 | 1  0 |  | p’() : @con.wait(&lock)  lock.release(); |  |
| 0  0  1  1 | 0  1  1  0 |  | V():  Lock.acquire();  Val++;  Lock.release(); |  |

**2. Semaphore로 Condition Variable을 구현 하시오. (Extra credit)**

**Basics**)

Condition variable : a queue of threads waiting for something inside a critical section

–Wait(&lock): Atomically release lock and go to sleep. Re-acquire lock later, before returning.

–Signal(): Wake up one waiter, if any

–Broadcast(): Wake up all waiters

**Building condition variable**)

Int numWaters;

Lock atom; //atomic operation을 위해서 씀

Semaphore waiter; //queue대신 쓸 semaphore. Init=0이고 주머니는 커야함

Semaphore wakers; //queue대신 쓸 semaphore. Init=0이고 주머니는 커야함

Wait(&Lock lock) {

Atom.acquire;

numWaters++;

lock.release();

atom.release; //여기까지 atomic

waiter.P();

lock.acquire();

}

signal() {

atom.acquire();

if(numWaiters>0) {

numWaiters--;

waiter.V();

}

Atom.release();

}

Broadcase() {

Atom.acquire();

While(numWaiters>0) {

Waiters.V();

NumWaiters--;

}

Atom.release();

}

**Flow example**)

Process A, Process B가 race condition상태에 있다고 할 때

|  |  |  |  |
| --- | --- | --- | --- |
| Sem  (init  =0) | Num  Waiter | A | B |
| 0  0  0  0  0  0 | 0  0  1  1  1  1 | Wait{  Atom.acquire;  numWaters++;  lock.release();  atom.release();  <-interrupt! |  |
| 0  0  0  1  1 | 1  1  0  0  0 |  | Signal{  atom.acquire();  numWaiters--;  waiter.V();  Atom.release(); } |
| 0  0 | 0  0 | waiter.P();  lock.acquire();} |  |
| //정상 작동됨 | | | |
| 0  0  0  0  0  0 | 0  0  1  1  1  1 | Wait{  Atom.acquire;  numWaters++;  lock.release();  atom.release;  waiter.P(); //기다림 |  |
| 0  0  0  1  1 | 1  1  0  0  0 |  | Signal{  atom.acquire();  numWaiters--;  waiter.V();  Atom.release(); } |
| 0  0 | 0  0 | waiter.P();  lock.acquire();} |  |
| //정상작동됨 | | | |

**과제가 너무 어려워요 살려주세요**