9/8/2010

Locks	ONLY	solve Mu	tual	Exclusion
				quencing problems
	another T			

Monitors	s have	3 components
£ 1. Lock	for mutu	al exclusion
(2. One,	or more, con	ndition variables sequencing /
() 3. Monito	r variables segui • just dat	for making ? encing decisions? ta
Shared	data	

Condition Variables Have no state Have 3 operations

Wait: Put myself to sleep, waiting on some condition

Signal: Wake up 1 waiting threadif there is one

Broadcast: Wake up all waiting threads

Form of a Monitor
Acquire a lock- "Entering the monitor
Access/manipulate our monitor
Access/manipulate our monitor variable(s) to make Synchronization decisions
Use C.V. operations to wait, signal, or broadcast
Release the lock - "Exiting the monitor"

Example: Do I get in "Line"? [int lineLength = 0; // monitor variable shared } Luck + lineLock = new Lock ("-"); data Condition + LineCV= new Condition ("-")
//To see if I get in line LineLock > Acquire (); if (lineLength > 0) { lineLength+ lineCV-> Wait (lineLock); LineLength
* // Not in line - place an order line Luck-> Release();

inside Condition Lock+ waiting Lock = NULL; void Condition: Wait Check+ lock) disable interrupts if (lock = = NULL) { print msg, restore interrupts, return if (waiting Lock == NULL) { 1/ First thread calling Wait, Save the waiting back = lock; // make sure the input lock matches

If the saved lock

if (waiting Lock) != lock) {

// Locks must match

print msg, R.I. & return

}

// Everything OK to be a waiter"

// Add thread to Condition wait Q

lock-7 Release();

current Thread >> Sleep();

lock-7 Acquire();

restore interrupts

```
void Condition: Signal (Lock+ lock) {
    disable interrupts
 * // If no waiters, restore interrupts & return
   # if (waiting Lock != lock) {
         print msg, R.I. & return
    // Wakeup 1 waiter
        · Remove 1 thread from Condition
        . Put them on Ready Q (@aback)
     if (/*no more waiting threads+/) {
waiting Lock = NULL;
     R.I.
```

void Condition: Broad cast (Lock + lock) {
while (/+ there are waiters +/) { Signal (lock);
3 3 19 nati C 102 L 3/

Lock+ lock1, lock2; Waiter Signaler lock1-> Acquire(); lock2-> Acquire(); Broadcast cv1-> Signal(lock2)) lock4->Acquire(); cv1-> Wait (lock1);

ABit More on Monitor Theory
Situation
Thread P is running in the CPU & the monitor
×
Thread Q, was in the monitor, but is now waiting on some condition
Pexecutes a Signal to wakeup Q
Probleme D& O count both he is

Problem:	P&Q	cannot	both 6	e u
	the n	nonitur @	the sa	metine
—				

0	nly Reol Wesa-style	Choice 3	Q for mon	has to use access to itor	sait the

Example: Producer / C	on sumer
"makes"	Y ()
an item	an item
use a monitor	2
use a monitor int itemCount = 0;	// Number of
	produced items
11 Assume an infinite	. buffer
Lock monitor Lock;	

Condition need I tem; // For Consumer to wait for an item

Producer() {

while (true) {

monitor Lock, Acquire();

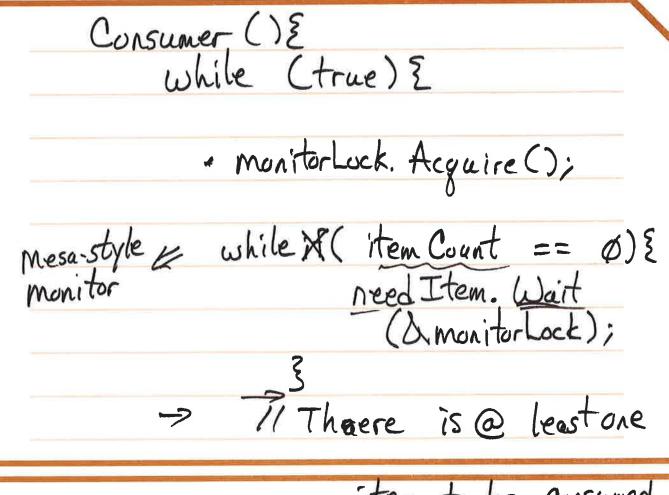
// Produce an item

// Put in buffer

item Count ++;

need Item, Signal
(& monitor Lock);

monitor Lock. Release ();



item to be consumed

// Consume 1 item

item Count --;

monitor Lock, Release();

3

