CMPSCI 377 Operating Systems **Fall 2005**

Lecture 7: October 11

Lecturer: Emery Berger *Scribe: Rob Tate, Lou Karadashkov*

Today:

- Implementing Locks
- Semaphores
- Bonus topics!

7.1 Implementing Locks

7.1.1 Atomic Instructions

Atomic instructions are provided by the architecture.
-Cannot be interrupted
-Needed to build locks
-Test and Set (most architectures)
-Exchange(x86)
-LLSC–Load Link Store Condition(PowerPC)

Example(this *must* be atomically provided by architecture):

```
int testset (int value)
{
    int old = value;
    value = 1;
    return old;
}
```

7.1.2 Using "Test and Set"

To acquire, spin in a while loop, like "while(testset(value)){"
To release, set value back to 0
-Busy-Loops are bad news.
-"Priority Inversion"
-High priority threads *might* get the lock, but they should *always* get it before lower priorities
-Will actually work
Locks should be optimized for the common case, that they are *uncontended*.

7.1.3 A Better Solution

Let's use a queue:

```
void acquire(){
    while(true){
        if (testset(value)){
            put thread on queue
            go to sleep
        }else{
            break;
        }
    }
}

void release(){
    value = 0;
    wake up threads
}
```

How many threads to wake up?
-If we wake up only one, we might get Priority Inversion
-If we wake them all up, we might get "Thundering Herd"
-We should use a priority queue and wake up the one at the front

Provides Mutual Exclusion, but...
-Deadlock can easily happen.
-Someone can forget to unlock.
-Someone can try to lock twice.

What if we want to let two people access but not three? Can't do this with locks. We need Semaphores!

7.2 Semaphores

7.2.1 What are Semaphores?

Semaphores are basically non-negative integer counters with atomic methods for incrementing and decrementing.
These methods are used to make wait and signal methods.

-wait() decreases semaphore by one unless it is zero, in which case it goes on queue
    -also called P()
-signal() increases semaphore by one and wakes up thread(s)
    -also called V()

- Think of semaphores as traffic lights that know how many cars should be allowed through.

- Semaphores are more general than locks.
  - Binary Semaphores are basicall the same thing as locks
  - Counting Semaphores are other semaphores, sometimes called "Dijkstra Style"

7.2.2 How do we use semaphores?

1) Use them just like locks
   -start with value of 1
   -mutual exclusion is maintained
2) Use them for resourse management
   -start with value that represents how many threads should be able to access resource
3) Use them as triggers
   -start with value of 0, signal() will trigger threads that have called wait()

7.3 Bonus Topics!

7.3.1 Multiprocessing

"Cache Coherance" is knowing the state of the memory copied into your cache.
   -found in SMP(symmetric Multi-Processing)
   -MESI is one common type
      -Modified, Exclusive, Shared, Invalid bits for each piece of cached memory
   -If two processors are competing, it's called cache line ping-pong. This is very expensive.

There is also NUMA (non-uniform memory access) systems:
   -each processor has some memory that is quick to access and some that is slow
   -easier and cheaper to build
   -*no cache coherance*

7.3.2 Edsger Dijkstra

- shortest path
- Algol-60, first modern compiler
  - had stack for implementing recusion
  - vectors
  - BNF, without which Fortran sent Voyager into the sun
- discovered "deadly embrace" (deadlock)
- invented semaphores