CS 350 Notes

Matthew Visser

Sep 22, 2011

1 Mutual Exclusion

- A thread can be preempted at any time in Peterson's algorithm.
- Peterson's algorithm only works for two threads.

1.1 Starvation

A thread may never get to execute because of the way that the synchronization algorithm or scheduler works.

Can use special instructions:

- Test-and-set
- Swap

1.2 Test-and-Set

Sets the value of a memory location and applies a condition on the old value. Since this is atomic, only one thread will execute it first, leaving the others with the new value.

1.3 Swap

Can swap and then test old value.

1.4 How do we wait until it's our turn?

You can either spin or block. Spinning is busy waiting, such as `while( condition ) ;`. Blocking is sleeping. If you don't expect to spin for long, then it may be better to spin than yield. Otherwise, if it will take a long time, it may be better to yeild.

2 Other Synchronization Primitives

2.1 Semaphores

- Can be used to enforce mutual exclusion requirements. It can also enforce other synchronization problems.
- Has integer value and supports two operations:
  - P → Wait — call this before starting the critical section.
  - V → Signal — call this after leaving the critical section.
- Can allow multiple threads to access a critical section, but limit the number.
- Need to enforce that `P()` and `V()` are atomic. If on a multi-core CPU, `P() V()` themselves are critical sections.

2.2 Thread Blocking in OS161

- `thread_sleep(const void *addr)` — sleeps the current thread.
- `thread_wakeup(const void *addr)` — wakes up all threads that are sleeping on `addr`.