CSci 4061: Recitation 12

April 25, 2016
Today: Synchronization Wrap-up

• Abstraction
  – (mutex) Locks
  – Condition variables
  – Semaphores

• Pitfalls
  – Deadlock
  – Race conditions
  – Livelock
  – Starvation
Synchronization: Abstractions

• Mutex locks
• Condition variables
• Semaphores
• *Keep in the back of your mind: do we need all three?*
Synchronization: Locks

- `pthread_mutex_t`
- `pthread_mutex_{try}lock`
- `pthread_mutex_unlock`
- Natural fit for
  - Enforcing exclusive access to CS
- Pitfalls
  - Lock too much/little
  - Forget to unlock after locking
  - Forget to use the lock at all
  - **Deadlock**, livelock, etc…
Synchronization: CVs

• Condition *variable*?
  – Do they really maintain any state that the user cares about?

• `pthread_cond_t`
• `pthread_cond_{timed}wait`
• `pthread_cond_{signal,broadcast}`
Synchronization: Semaphores (1/5)

• State
  – count, queue
  – count >= 0 → queue is empty

• Up / Signal / V
  – Unblock a waiter, or if none, then increase count

• Down / Wait / P
  – Decrease the count, or if already 0, add the caller to wait queue
Synchronization: Semaphores (2/5)

- semaphore.h
- sem_t
- sem_init
- sem_{try,timed}wait
Synchronization: Semaphores (3/5)

• Implement semaphore using locks?

```python
wait():
    keep_waiting = true
    while keep_waiting:
        lock.acquire()
        if count > 0
            count--
            keep_waiting = false
        lock.release()
```

• What would signal do?
• How can we avoid this busy-waiting?
Synchronization: Semaphores (4/5)

- Implement semaphore using locks... and CVs?

```python
wait():
    lock.acquire()
    while count <= 0:
        cond.wait(lock)
    count--
    lock.release()
```

- What about signal?
- Why `while`, and not `if`?
- Why do we need both a CV and a lock?
Synchronization: Semaphores (5/5)

• How is a semaphore different from an integer?
• How is it different from a mutex lock?
  – Pros/cons?
  – Implement a mutex lock using semaphores?
• Implement a condition variable using semaphores?
• …so do we really need all three?
Deadlock: Trivial Example

• Thread A
  – lock(x)
  – lock(y)
• Thread B
  – lock(y)
  – lock(x)
• Problem?
Deadlock: Conditions

- Mutual exclusion
- Hold-and-wait
- No preemption
- Circular wait

- Too abstract? Let’s philosophize.
Deadlock: Solutions

• Bury your head in the sand?
• Order locks
  – What condition does this avoid?
  – Can you think of any drawbacks?
  – philosophize_ordered.c
• Trylock
  – Same questions
  – philosophize_trylock.c
• Other ideas?
Questions?