Assignment 2
More on Synchronization

Ari Rabkin

First, an apology

- I had intended to give you all some examples and templates of design docs.
- I didn't get to it until Sunday. I'm sorry about that.
- We'll try to do it earlier next time.
- Remember, TAs are students too!

Congratulations!

- Most of the designs I saw were quite good.
  - One clarification: don’t regurgitate skeleton code or lectures.
- Project One turned out really well.
  - Median and Mean were in the 80s

Synchronization

- Synchronization is a key theme of 414/415.
- Crucial for thread safety (and OS design).
- Unavoidable in distributed systems.
- Project Two is preemptive...watch out!
- Severe grade penalties for thread unsafety.

Two uses:

- Two uses of synchronization primitives:
    - Inter-thread control flow (signaling)
    - Concurrency control without explicit dependence (locking)
- Use semaphores for both.

Locking: why

- Suppose some data structure is shared between threads.
- If both threads update concurrently, can corrupt values.
- Scheduler dependent, hard to debug -- heisenbugs are hard to track down.

Locking: how

- Two ways of locking in minithreads:
- Can use a semaphore or disable interrupts.
- Use semaphores when you can, turn off interrupts when you must.

Locking: where

- Need to do this for every data structure shared between threads.
- If two threads can access concurrently, generally need locks or atomic operations.
- For you, locks are usually the way to go.
- Check with us if you want something fancy.

Synchro != Locking

- Locking isn’t only synchro issue.
- Often are more subtle race conditions. Watch out!
- Also, beware deadlock

Don’t “Roll Your Own”

- Temptation to hack scheduler to do magic.
- Resist this!! Your code will be severely penalized, and also irreparably broken.
- Use semaphores for all inter-thread synch.
- Check with us if you think you have a special case.

Windows API

- Don’t make windows API calls without checking with us first.

The idle thread

- Idle thread stays running
- Just does while(1) ; for this project.
- yield() disables interrupts -- bad

“Security”

- Minithreads is a thread package, not an OS.
- Threads aren’t protected from each other.
- Not meaningful to worry about malicious threads: can’t protect against them anyway.
- But good to catch programmer error.

Wall vs CPU time

- Thread running and getting interrupts: all counters tick.
- Thread running (interrupts off): cpu and wall clocks tick
- Thread “running”, but minithreads isn’t: wall clock ticks, no others
- Thread not running: all counters frozen

Use