Chapter 5 – Deitel – Lesson Plan for Abrams' Section

Items to Highlight

Section 5.1

- Topic of CH5 is how to implement synchronization .primitives
- (Ch6: is how to use those primitives to write useful programs)

Section 5.2

- Consider again web server example.
- To prevent denial of service attack, we used a counter to limit number of forked children.
- How is "CTR++" implemented ATOMICALLY?
    - Harder than it first appears!
    - Assembly sequence: Load, INC, Store
    - What if multiple children interleave this sequence?
        - If CTR=100, and 2 children do sequence, what are all possible values of CTR after both finish CTR++? (101 or 102)
    - HOW CAN WE FORCE SERIALIZATION?
        - Either do it in SOFTWARE or in HARDWARE to LOCK and UNLOCK before/after CRITICAL SECTION.
        - Chapter 5 has software algorithms.
        - Hardware solution might be EXCHANGE instruction.
        - Key is to do load/store atomically.

Section 5.2.1 Java Code for producer/consumer with NO synchronization

- SLIDE 16: the main program launches producer & consumer
- SLIDE 17-19: possible outputs
- Classes: Unsynchronized, Producer, Consumer
    - Producer sets buffer successively to 1 to 4.
    - Consumer reads buffer 4 times and outputs value.
- Interface: Buffer (Slide 7 and then Slides 14-15 with implementation)
- Producer: Slides 8-10
- Consumer: Slides 11-13
- WHAT IS MAX VALUE consumer will ever print? 4*4=16/
- WHAT IS MIN VALUE consumer will ever print? 4*(-1)=-4.

Section 5.4: Software Solution to ME

Let's play a game. Two teams. Each can read/write a value that a 3rd player (memory) keeps track of. How can we guarantee two teams are not simultaneously in classroom?

Solution: Dekker's algorithm (slides 24-25)

Section 5.4.3: Lamport's Bakery Algorithm (slides 47-50)

Purpose: Mutual exclusion among N threads

Compared to Dekker & Peterson algorithms:

- Works with N, not 2, threads
- Works with multiprocessors
- Overview:
  - Tickets assigned like a bakery.
  - However, 2 tickets can have same value.
  - Priority given to LOWEST ticket number AND SMALLEST thread number to break ties.
- Look at algorithm on slides 48-51

Good properties of the algorithm:

- Guarantees ME
- No indefinite postponement
- No deadlock

Section 5.5: HW Solutions

1. Disable Interrupts (what are pros/cons? Does it work on a multiprocessor?)
2. TestAndSet (or ReadModifyWrite)
3. Swap

Section 5.6: Semaphores

- IMPORTANT: Semaphore is a programming concept that is built on some ME mechanism (Lamport bakery, test/set, swap, disable interrupts)
- Semaphore = Non-negative integer + FIFO queue of waiting threads
- Two types of semaphores
  - Binary (value is 0 or 1)
  - Counting (value is integer)
- Two operations
  - P (decrement or block)
  - V (unblock or increment)
- P operation:
  - if S>0 then S-- else block
- V operation:
  - if Q not empty then unblock else S++
- Go over slides 61-68