CSCE 313-200
Introduction to Computer Systems
Spring 2018

Practice
Dmitri Loguinov
Texas A&M University

February 20, 2018
**Midterm #1**

- **Problem 6**
  - Part in bold is important to solving this question
  - Upper bound is sequential execution, we get 100
  - Lower bound?
- **Problem 2: WRR queuing**
  - If N jobs are pushed on the CPU, \((1-w)N\) are from low-priority Q
  - Equating \(K = (1-w)N\), we obtain \(N = K / (1-w)\) time slices
  - Total delay = \(N\Delta\) seconds where \(\Delta\) is one slice delay

```c
for (int count = 0; count < 50; count++)
    tally ++;
// converted by compiler to
// __asm {
    // mov   eax, tally
    // inc   eax
    // mov   tally, eax
// }
```

```c
mov   eax, 0
// switch
```

```c
mov   eax, 0
inc   eax (=1)
mov   tally, eax
 mov   eax, 1
...  
mov   tally, 49
// switch
```

```c
mov   eax, 1
// switch
```

```c
inc   eax, (=2)
mov   tally, 2
```

- high-priority queue always non-empty
- K low-priority processes already in queue
Semaphore Problems

• Concurrency is a difficult concept  
  − Cannot be understood without practice
• Threads are replaced with arbitrary actors  
  − E.g., “no more than 15 animals can enter the room”
• Rules for semaphore/mutex solutions
• 1) All wait() functions are blocking  
  − No timeouts to break out of deadlocks
• 2) No looping while waiting for events  
  − Example on the right is not acceptable →
• 3) Bulk semaphore release(N) is available
• 4) Semaphore release beyond max throws an error

```plaintext
mutex.Lock();
while (Q.size() == 0)
  mutex.Unlock()
  Sleep
mutex.Lock()
```
Semaphore Problems

• In programs, you can obviously violate these rules
  - However, tests will require less-straightforward approaches that demonstrate your grasp of synchronization theory

• Exam preparation guide:
  - Little Book of Semaphores
  - http://greenteapress.com/semaphores/

• Make sure to actively attempt solving problems
  - Tests will have similar levels of difficulty

• Problem #1
  - Bears and goats come to a party; however, the barn can hold only 15 animals max

```c
void EnterBarn (void) {
    // called when animal wants to enter
}

void Party (void) {
    // called when partying
}
Semaphore Problems

- **Problem #2**
  - Barn holds no more than 8 bears and no more than 12 goats at any time

- **Problem #3**
  - No more than 8 bears, no more than 12 goats, and no more than 15 combined

- **Problem #4**
  - First animal to enter turns on the lights
  - Last animal to exit turns off lights
  - Nobody can enter or leave while lights are being manipulated

- **Problem #5**
  - If Pig (assumed to be unique) shows up to party, no other animal can enter until Pig voluntarily leaves

```c
void EnterBarn (int type) {
    // 0 = goat, 1 = bear
}

void LeaveBarn (int type) {
    // 0 = goat, 1 = bear
}

void TurnOnLights(void) {
    // gets called if room is dark
}
```
Semaphore Problems

• Problem #6
  - Pig wants to crash the party, but with style
  - If Pig arrives and fewer than 50 animals are in barn, it waits
  - While Pig is waiting, new animals may enter or depart; once critical mass of 50 is reached, the pig crashes party
  - While Pig is inside, all arriving animals must wait outside until Pig departs

• Problem #7
  - Same as #6, but Pig locks the door, nobody can leave

• Problem #8
  - If room is empty, any animal may enter
  - If room has someone inside, new animals must wait outside until they are allowed to enter by whoever is departing
  - Departing animal prefers to let animals of the same type in
Semaphore Problems

• Work on these at home
  - Were on the test last year

• Problem #9
  - Bears and goats come to party at the barn; main caveat is bears may get drunk and start eating goats
  - If barn is empty, either type of animal may enter
  - If bears are inside, arriving bears should enter without delay
  - If goats are inside, arriving goats should enter without delay

• Problem #10
  - Same as #9, but barn occupancy is 50 animals max

• Problem #11
  - Same as #9, but ensures lack of starvation