Semaphore Solutions

Semaphore

- Briefly describe the semaphore data structure
  - A semaphore can be used to synchronize threads
  - It has a non-negative integer as a data member ("the counter")
  - Calling the acquire() member function decrements the counter
  - Calling the release() member function increments the counter
  - If the counter is equal to 0, acquire() blocks until the counter becomes non-zero
  - The counter cannot be incremented beyond some maximum value

Simple Semaphore Implementation

- Write a simple implementation of a semphore
  - You may use a mutex and a condition variable in your class
- Write a program to test your implementation

Binary Semaphore as Mutex

- What is a binary semaphore?
  - In a binary semaphore, the counter can only have the values 0 and 1
- Describe how a binary semaphore can used as a mutex
  - A thread entering a critical region calls acquire()
  - The semaphore's counter is decremented to 0
  - If it succeeds, all other threads which call acquire() will be blocked, because they cannot decrement the counter
  - When the thread leaves the critical region, it calls release()
  - The semaphore's counter is incremented to 1
  - This will unblock one of the other threads which called acquire()
  - That thread can now decrement the counter and enter the critical region

Binary Semaphore as Condition Variable

- Describe how a binary semaphore can used as a condition variable
  - A thread waiting for a signal calls acquire()
  - The semaphore's counter is decremented to 0
  - This thread sleeps until the counter is incremented
  - The notifying thread calls release()
  - The semaphore's counter is incremented to 1
  - The waiting thread wakes up and resumes execution
- How can we notify more than one thread?
  - Use a semaphore where the maximum counter value is equal to the number of threads to be notified

Semaphores

- What are the advantages of semaphores?
  - Semaphores are more flexible. They can notify any given number of waiting threads
  - Simpler code. They avoid working with mutexes and condition variables
  - The performance can often be better
  - Semaphores are more versatile. They can be used to create more complex synchronization objects