More threads

2/8/16
What is a condition variable?
void insert(Item x) {
    pthread_mutex_lock(&lock);
    while(buffer_is_full()) {
        pthread_cond_wait(&nonfull, &lock);
    }
    add_to_buffer(x);
    pthread_cond_signal(&nonempty);
    pthread_mutex_unlock(&lock);
}

Item remove() {
    pthread_mutex_lock(&lock);
    while(buffer_is_empty()) {
        pthread_cond_wait(&nonempty, &lock);
    }
    Item x = remove_from_buffer();
    pthread_cond_signal(&nonfull);
    pthread_mutex_unlock(&lock);
    return x;
}
void insert(Item x) {
    pthread_mutex_lock(&lock);
    while(buffer_is_full()) {
        pthread_cond_wait(&nonfull, &lock);
    }
    add_to_buffer(x);
    pthread_cond_signal(&nonempty);
    pthread_mutex_unlock(&lock);
}

Item remove() {
    pthread_mutex_lock(&lock);
    while(buffer_is_empty()) {
        pthread_cond_wait(&nonempty, &lock);
    }
    Item x = remove_from_buffer();
    pthread_cond_signal(&nonfull);
    pthread_mutex_unlock(&lock);
    return x;
}

Why is the lock passed to pthread_cond_wait?

A. The condition variable is used for signaling, but the thread blocks on the lock
B. So the lock can be released before blocking
C. Allows a more efficient library implementation
D. Reminder to the programmer that condition variables must be protected by locks
E. More than one of the above
What is the purpose of the cleanup stack?

A. Store low-priority tasks to be completed on a best-effort basis
B. Allow the main threads to defer memory deallocation and file closure operations
C. Let a terminated thread return the system to a consistent state
D. To distinguish dirty dishes from clean ones
E. Not exactly one of the above
What is the purpose of the cleanup stack?

A. Store low-priority tasks to be completed on a best-effort basis
B. Allow the main threads to defer memory deallocation and file closure operations
C. Let a terminated thread return the system to a consistent state
D. To distinguish dirty dishes from clean ones
E. Not exactly one of the above
What is priority inversion?

A. When the programmer wants to change thread priorities due to a phase transition in the program

B. When the scheduler changes a thread’s programmer-assigned priority

C. When one thread ends up waiting for a thread with lower priority

D. When it’s really important to flip something over

E. None of the above
What is priority inversion?

A. When the programmer wants to change thread priorities due to a phase transition in the program
B. When the scheduler changes a thread’s programmer-assigned priority
C. When one thread ends up waiting for a thread with lower priority
D. When it’s really important to flip something over
E. None of the above
Java and OpenMP
When using threads in Java, which of the following is NOT an advantage of implementing Runnable rather than extending Thread?

A. Leaves open the possibility of inheriting from another class
B. Allows use in other frameworks rather than just running as a separate thread
C. Makes it clear that core Thread functionality is not being changed
D. Does not require writing a pause method
E. All of the above are advantages of implementing Runnable
When using threads in Java, which of the following is NOT an advantage of implementing Runnable rather than extending Thread?

A. Leaves open the possibility of inheriting from another class
B. Allows use in other frameworks rather than just running as a separate thread
C. Makes it clear that core Thread functionality is not being changed
D. Does not require writing a pause method
E. All of the above are advantages of implementing Runnable
Synchronized methods in Java are an implementation of what parallel programming construct?

A. Lock / mutex
B. Condition variable
C. Semaphore
D. Monitor
E. Not exactly one of the above
Synchronized methods in Java are an implementation of what parallel programming construct?

A. Lock / mutex
B. Condition variable
C. Semaphore
D. Monitor
E. Not exactly one of the above