Concurrent Queue with Condition Variable Solutions

Concurrent Queue

- The queue from the previous lecture is safe
  - pop() throws an exception when the queue is empty
  - push() throws an exception when the queue is full
- Suggest how it could be made more useful
  - pop() waits until there is some data on the queue
  - push() waits until the queue is no longer full

Concurrent Queue with Condition Variable

- Describe how you could do this with a condition variable
  - The thread that calls pop() calls wait() on the condition variable
  - The thread that calls push() notifies the condition variable
- wait() will be called on the condition variable
- Would it be useful to add a predicate argument to this call?
  - Yes, to avoid spurious and lost wakeups
- What would this predicate do?
  - If the queue is empty, we continue waiting
  - If it is not empty, then it is safe to continue and pop from the queue

concurrent_queue with Condition Variable

- Implement your solution
- Write a multi-threaded program to exercise your implementation
- Check that it behaves correctly, including the following cases:
  - pop() called when the queue is empty
  - push() called when the queue is full

Conclusion

- This is a simple concurrent queue implementation
- Describe its limitations
    - It employs "coarse-grained" locking
    - Only one thread can access the queue at any one time
    - In effect, the program becomes single-threaded
- Does adding the condition variable improve the situation?
    - It has slightly more concurrency
    - If the queue is empty and a thread is trying to pop()
    - Other threads can run, until the queue is no longer empty
- Are there other ways to implement a concurrent queue?
    - A lock-free solution would be more efficient
    - But much more complex