Condition Variables Solutions

Condition Variable Scenario

- In this scenario, the processing thread creates a unique_lock instance to lock the mutex, but the fetching thread creates a lock_guard
- Why do the two threads use different lock types?
  - The processing thread needs to be able to unlock the mutex. This is done inside the wait() call. In some applications, the waiting thread may need to directly unlock the mutex as well.
  - Therefore the mutex must support unlock()
  - Hence unique_lock is used

Condition Variable Scenario

- The fetching thread only needs the mutex to protect the critical section. The extra flexibility and overhead of unique_lock is not needed here, so we use the simpler lock_guard
- We put the lock_guard and the critical section in their own scope
- The mutex will be automatically unlocked by the lock_guard destructor

Condition Variable Example

- Now reverse the order of the threads, so that the writer thread is started first. Add a sleep (say, half a second) before starting the reader thread. What happens?
  - The writing thread completes before the reading thread runs
  - The reading thread has not called wait() on the condition variable
  - The condition variable does not have any waiting threads to notify
  - The notification is "lost"
  - By the time the reading threads calls wait(), the writing thread has completed
  - The condition variable does not receive any more notifications
  - The reading thread blocks for ever