Condition Variables Solutions

Condition Variable Motivation

- Write a program which starts two threads
  - A "writer" thread which modifies a shared string
  - A "reader" thread which uses the string, only after it has been modified
- Use a condition variable to coordinate the threads
- Avoid data races

Condition Variable Scenario

- In the example in the video
  - The reader thread uses std::unique_lock
  - The writer thread uses std::lock_guard

Condition Variable Scenario

- Why do the two threads use different types?
  - The reader 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)
  - The mutex must support unlock()
  - Hence std::unique_lock is used in the reader
  - The writer thread only needs the mutex to protect the critical section
  - The extra flexibility and overhead of std::unique_lock are not needed here
  - The writer uses the simpler std::lock_guard

Condition Variable Example

- Now reverse the order of the threads in main()
- The writer thread is started first
- Add a sleep (say, half a second) before starting the reader thread
- What happens, and why?
    - The writer thread completes before the reader thread runs
    - The reader 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 reader thread calls wait(), the writer thread has completed
    - The condition variable does not receive any more notifications
    - The reader thread blocks for ever