Condition Variable

used to signal between threads. [thread(s) block until signaled]

overview

bool isReady = false; // global

T1 (waiting on isReady == true) ← blocked in meantime

need some locking

T2 (sets isReady = true & then notifies any other threads waiting.

std::condition_variable

Notifying

notify_one → OS wakes 1 thread (can't tell which one)

notify_all → OS wakes them all (1 will acquire mutex & continue) the rest wait until they get mutex & continue

Waiting

wait — wait until notified

wait_for, wait_until } timed versions of above ignore for now

```
#include <mutex>
#include <condition_variable>

mutex m;
condition_variable c;
bool ready = false;
void get() {
    unique_lock<mutex> lck(m);   <- acquire
    while (!ready)
        c.wait(lck)              <- release & wait for signal
}

void set() {
    unique_lock<mutex> lck(m);
    ready = true;                // all access to ready protected
    cv.notify_all;               // wake up all other threads.
}
```

unique_lock verses lockguard?
lockguard is locked until destruct

unique_lock is all that u can unlock, create unlocked then lock
unique_lock a little heavier, always use lock_guard unless you need unique lock abilities