A helper template function to wait for a Win32 condition variable in a loop

devblogs.microsoft.com/oldnewthing/20180119-00

January 19, 2018

Raymond Chen

Win32 condition variables suffer from the problem of spurious wake-ups, and you will usually wait on a condition variable in a loop. It's easier than the case of `WaitOnAddress` because you hold a lock while checking the condition, so you don't have to worry about race conditions against other threads. (The idea is that anybody who wants to cause a change to the condition needs to acquire the same lock. Therefore holding the lock prevents the condition from changing.)

The C++ standard library contains an overload to the `std::condition_variable::wait` method which takes a predicate. If the predicate returns `false`, then the `wait` will loop back and wait some more.

Let's write that same helper function for Win32 condition variables.

```
template<typename TLambda>
void SleepConditionVariableCSUntil(
  CONDITION_VARIABLE* conditionVariable,
  CRITICAL_SECTION*   criticalSection,
  TLambda&&           is_okay)
{
  while (!is_okay()) {
    SleepConditionVariableCS(conditionVariable, criticalSection, INFINITE);
  }
}

template<typename TLambda>
void SleepConditionVariableSharedSRWUntil(
  CONDITION_VARIABLE* conditionVariable,
  SRWLOCK*            srwLock,
  TLambda&&           is_okay)
{
  while (!is_okay()) {
    SleepConditionVariableSRW(conditionVariable, srwLock, INFINITE,
                              CONDITION_VARIABLE_LOCKMODE_SHARED);
  }
}

template<typename TLambda>
void SleepConditionVariableExclusiveSRWUntil(
  CONDITION_VARIABLE* conditionVariable,
  SRWLOCK*            srwLock,
  TLambda&&           is_okay)
{
  while (!is_okay()) {
    SleepConditionVariableSRW(conditionVariable, srwLock, INFINITE, 0);
  }
}
```

I will admit that these helpers aren't as useful as the one for `WaitOnAddress` because the loop is very straightforward. It may not be much of a benefit over just writing the loop out manually.

Raymond Chen

Follow