Extending our critical section based on WaitOnAddress to support timeouts

devblogs.microsoft.com/oldnewthing/20170531-00

May 31, 2017

Raymond Chen

Let's take the critical section we constructed in terms of `WaitOnAddress` and add two new functions:

- `TryEnterAltCs` tries to enter the critical section if it is either available or is already owned by the current thread. If the critical section is owned by another thread, then the call fails.
- `TryEnterAltCsWithTimeout` which tries to enter the critical section but gives up after waiting for the specified timeout.

The first function is a simplification of the existing `EnterAltCs` function. It simply gives up if the critical section is not available.

```
bool TryEnterAltCs(ALTCS* Cs)
{
  DWORD ThisThread = GetCurrentThreadId();
  DWORD PreviousOwner = InterlockedCompareExchangeAcquire(
              &Cs->OwnerThread, ThisThread, 0);
  if (PreviousOwner == 0) {
    Cs->ClaimCount++;
    return true;
  }

  if (PreviousOwner == ThisThread) {
    Cs->ClaimCount++;
    return true;
  }

  return false;
}
```

The second function is a modification of the existing `EnterAltCs` function that gives up after waiting too long:

```
// Timeout is in milliseconds and cannot be INFINITE.
bool TryEnterAltCsWithTimeout(ALTCS* Cs, DWORD Timeout)
{
  ULONGLONG Deadline = GetTickCount64() + Timeout;

  DWORD ThisThread = GetCurrentThreadId();
  DWORD TimeRemaining;
  do {
    DWORD PreviousOwner = InterlockedCompareExchangeAcquire(
               &Cs->OwnerThread, ThisThread, 0);
    if (PreviousOwner == 0) {
      Cs->ClaimCount++;
      return true;
    }

    if (PreviousOwner == ThisThread) {
      Cs->ClaimCount++;
      return true;
    }

    ULONGLONG Now = GetTickCount64();
    if (Now >= Deadline) {
      return false;
    }

    TimeRemaining = static_cast<DWORD>(Deadline - Now);
  } while (WaitOnAddress(&Cs->OwnerThread,
      &PreviousOwner, sizeof(PreviousOwner), TimeRemaining));
  return false;
}
```

The changes we made were to keep track of how much time remains before the deadline. If the deadline passes, then we return `false`. Otherwise, we ask `WaitOnAddress` to wait for the owner to change, or for the timeout to elapse. The function returns `FALSE` if it returned due to a timeout, in which case we break out of the loop and return `false`. Otherwise, we were signaled (possibly spuriously), so we go back and try to claim the critical section again.

Raymond Chen

Follow