Creating other types of synchronization objects that can be used with co_await, part 4: The manual-reset event

devblogs.microsoft.com/oldnewthing/20210312-00

March 12, 2021

Raymond Chen

Now that we've finished our library for building awaitable synchronization objects, let's actually use it.

The introduction to this part of the series began with a demonstrate of a one-shot event. So let's take it a step further and make it a resettable event.

```
struct awaitable_manual_reset_event_state :
    async_helpers::awaitable_state<awaitable_manual_reset_event_state>
{
    awaitable_manual_reset_event_state(bool initial)
    : signaled(initial) {}

    std::atomic<bool> signaled;

    bool fast_claim(extra_await_data const&) const noexcept
    {
        return signaled.load(std::memory_order_acquire);
    }

    bool claim(extra_await_data const&) const noexcept
    {
        return signaled.load(std::memory_order_relaxed);
    }

    void set(node_list& list) noexcept
    {
        signaled.store(true, std::memory_order_relaxed);
        resume_all(list);
    }
};

struct awaitable_manual_reset_event
    : async_helpers::awaitable_sync_object<
        awaitable_manual_reset_event_state>
{
    awaitable_manual_reset_event(bool initial = false) :
        awaitable_sync_object(initial) { }

    void set() noexcept
    {
        action_impl(&state::set);
    }

    void reset() noexcept
    {
        get_state().signaled.store(false,
            std::memory_order_release);
    }
};
```

The manual reset event is basically the same as the one-shot event, except that you can also specify the initial signal state, and you can reset it as well as set it. The reset is done with release semantics, so that anything which was dependent upon the release will observe the changes that occurred prior to the release. (I'm not sure how you could actually rely on this, since the release doesn't signal anything, and all awaits will block on an unsignaled event, but I'm doing it just in case.)

Well, that was a bit anticlimactic. That's okay, we'll make up for it next time, when we look at auto-reset events.

Raymond Chen

Follow