C++ coroutines: Converting among tasks that use the same promise

devblogs.microsoft.com/oldnewthing/20210503-00

May 3, 2021

Raymond Chen

If the only difference between tasks is in the awaiter, then it's possible to convert between them without the promise even knowing what happened. We have an example of this with our `simple_task` and `com_simple_task`, which differ only in the awaiter produced by the `co_await` operator. This means that we can actually convert between the two by simple wrapping the promise inside the other class:

```
template<typename T>
struct simple_task : details::simple_task_base<T>
{
    using base = details::simple_task_base<T>;
    simple_task() noexcept = default;
    simple_task(details::simple_promise<T>*
        initial) noexcept : base(initial)
        { this->promise->start(); }
    simple_task(com_aware_task<T>&& other) noexcept
        : base(std::move(other)) { }
    ...
};

template<typename T>
struct com_aware_task : details::simple_task_base<T>
{
    using base = details::simple_task_base<T>;
    com_aware_task() noexcept = default;
    com_aware_task(details::simple_promise<T>*
        initial) noexcept : base(initial)
        { this->promise->start(); }
    com_aware_task(simple_task<T>&& other) noexcept
        : base(std::move(other)) { }
    ...
};
```

You can now take a `simple_task<T>` and re-wrap it inside a `com_aware_task<T>`:

```
extern async_helpers::simple_task<void> SomethingAsync();

auto task = com_aware_task<void>(SomethingAsync());
```

The `SomethingAsync` function returned a `simple_task<void>`, but we converted it to a `com_aware_task<void>`.

We can also do the same thing to convert a cold-start task to a simple task or com-aware task by adopting the promise and starting it. However, we cannot convert a hot-start task into a cold-start task because the task has already started; you can't un-start a task.

The last step here is to remove the need to retype the coroutine return value when performing the conversion. We do this by adding deduction guides.

```
    template<typename T>
    simple_task(com_aware_task<T>&&) -> simple_task<T>;
    template<typename T>
    com_aware_task(simple_task<T>&&) -> com_aware_task<T>;
```

Now you can write

```
auto task = com_aware_task(SomethingAsync());
```

Next time, I'll look at a dark corner of the coroutine specification and how danger lurks inside.

Raymond Chen

Follow