Hidden constraints on the result type in Concurrency Runtime tasks

devblogs.microsoft.com/oldnewthing/20200812-00

August 12, 2020

Raymond Chen

If you are using the Concurrency Runtime `task<T>` to represent asynchronous activity, there are some hidden constraints on the type `T`. If you violate these constraints, the compiler will complain, but perhaps not in an obvious way.[1]

If you try to create a `task<T>` where `T` does not have a public default constructor, then you get an error like

```
pptasks.h: error C2280:
'Concurrency::details::_ResultHolder<_ReturnType>::_ResultHolder(void)': attempting
to reference a deleted function
with _ReturnType = T
```

And if the `T` is not copyable, then you get something like

```
pptasks.h: Error C2280: 'T::T(const T&) noexcept' attempting to reference a deleted
function
```

What's going on?

Let's look at the copyability first. Task results must be copyable because the task result can be consumed multiple times in multiple ways. You can `get()` multiple times, and each time returns the task result. You can call `then()` multiple times, and each continuation is given the task result. If the task result were not copyable, then only one of the calls to `get()` or `then()` will get the result, and the others would get, um, a letter of apology?

The requirement that the type be publicly constructible is a consequence of the fact that the task contains a copy of the `T`, and if the task hasn't completed yet, then the `T` object needs to contains *something*, so the library just puts a default-constructed `T` in it. This requirement is confessed in the source code:

```
    // this means that the result type must have a public default ctor.
    _ResultHolder<_ReturnType> _M_Result;
```

Okay, so how can you work around these limitations? We'll look at that next time.

¹ **Related reading**: Why does my C++/WinRT project get errors of the form "`abi<…>::…` is abstract see reference to `producer<…>`"?, on the underappreciated need for libraries to generate comprehensible error messages when used incorrectly, and the difficulty of compiler error message meta-programming. For these two cases, the Concurrency Runtime could have added some `static_assert`s to generate custom error messages.

```
template<typename _Type>
struct _ResultHolder
{
    static_assert(std::is_default_constructible<_Type>::value,
        "The result type of a task must be default constructible");
    static_assert(std::is_copy_constructible<T>::value,
        "The result type of a task must be copy constructible");

    ....
};
```

Raymond Chen

Follow