How can I determine in a C++ header file whether C++/CX is enabled? How about C++/WinRT?

devblogs.microsoft.com/oldnewthing/20190610-00

June 10, 2019

Raymond Chen

Suppose you're writing a header file that wants to take advantage of C++/CX or C++/WinRT features if the corresponding functionality is available.

```
// async_event_helpers.h

#if (? what goes here ?)

// RAII type to ensure that a C++/CX deferral is completed.

template<typename T>
struct ensure_complete
{
   ensure_complete(T^ deferral) : m_deferral(deferral) { }
   ~ensure_complete() { if (m_deferral) m_deferral->Complete(); }

  ensure_complete(ensure_complete const&) = delete;
  ensure_complete& operator=(ensure_complete const&) = delete;

  ensure_complete(ensure_complete&& other)
  : m_deferral(std::exchange(other.m_deferral, {})) { }
  ensure_complete& operator=(ensure_complete&& other)
  { m_deferral = std::exchange(other.m_deferral, {}); return *this; }

private:
   T^ m_deferral;
};
#endif

#if (? what goes here?)

// RAII type to ensure that a C++/WinRT deferral is completed.

template<typename T>
struct ensure_complete
{
   ensure_complete(T const& deferral) : m_deferral(deferral) { }
   ~ensure_complete() { if (m_deferral) m_deferral.Complete(); }

  ensure_complete(ensure_complete const&) = delete;
  ensure_complete& operator=(ensure_complete const&) = delete;

  ensure_complete(ensure_complete&&) = default;
  ensure_complete& operator=(ensure_complete&&) = default;

private:
   T m_deferral{ nullptr };
};
#endif
```

What magic goes into the `#if` statement to enable the corresponding helpers only if the prerequisites have been met?

For C++/CX, the magic incantation is

```
#ifdef __cplusplus_winrt
```

If C++/CX is enabled, then the `__cplusplus_winrt` symbol is defined as the integer 201009, which is presumably a version number.

For C++/WinRT, the magic symbol is

```
#ifdef CPPWINRT_VERSION
```

This is defined to a string literal representing the version of C++/WinRT that is active. In addition to serving as a feature detector, this macro is used to ensure that all of the C++/WinRT header files you use are compatible with each other. (If not, you will get a compile-time assertion failure.)

The C++/WinRT team cautions that this is the *only* macro in the C++/WinRT header file that is supported for feature detection. Do not rely on the other `WINRT_*` macros in the C++/WinRT header files. They are implementation details and may change at any time.

Raymond Chen

Follow