Making the COM_INTERFACE_ENTRY macro better at detecting misuse

devblogs.microsoft.com/oldnewthing/20190905-00

September 5, 2019

Raymond Chen

Last time, we studied a crash involving improper use of the `COM_INTERFACE_ENTRY` macro. Can we make it better at detecting misuse at compile time?

We want to make sure that `IWidgetProviderInfo` derives directly from `IUnknown`, so that, given the declaration

```
IWidgetProviderInfo* p;
```

we can guarantee that `IWidgetProviderInfo` is convertible to `IUnknown` and that

```
reinterpret_cast<IUnknown*>(p) == static_cast<IUnknown*>(p)
```

Otherwise, somebody could try this:

```
struct Strange { ... };
interface IWidgetProvider : Strange, IUnknown
{
  ...
};
```

The `IWidgetProvider` can be converted to a `IUnknown`, but it will probably also need to be adjusted by `sizeof(Strange)`.[1]

ATL already has a macro for calculating the amount by which the two parts of the above comparison differ. It uses it for deciding how much to adjust the pointer to get to the desired interface:

```
#define offsetofclass(base, derived) \
    ((DWORD_PTR)(static_cast<base*>((derived*)_ATL_PACKING)) \
                                          -_ATL_PACKING)
```

So we just need to assert that the value is (1) calculable, and (2) zero.

The original definition of `COM_INTERFACE_ENTRY` is

```
#define COM_INTERFACE_ENTRY(x)\
    {&IID_##x, \
    offsetofclass(x, _ComMapClass), \
    _ATL_SIMPLEMAPENTRY},
```

We can make this minor adjustment to ensure that converting from `x` to `IUnknown` is both possible and a nop:

```
#define COM_INTERFACE_ENTRY(x)\
    {&IID_##x, \
    offsetofclass(x, _ComMapClass)/!offsetofclass(IUnknown, x), \
    _ATL_SIMPLEMAPENTRY},
```

If `x` cannot be converted to `IUnknown`, then the `offsetofclass` will encounter a compile-time error because the `static_cast` from `x*` to `IUnknown*` is not possible.

If the conversion is possible but requires a pointer adjustment, then the `offsetof` will produce a nonzero value. Negating it with `!` will produce `0`, and dividing by it will trigger a divide-by-zero compile-time error.

If the conversion is possible and does not require a pointer adjustment (which is the case we want to permit), then the `offsetof` will result in the value `0`. Negating it with `!` will produce `1`, and dividing by `1` has no effect.

You can apply this same fix to the other `COM_INTERFACE_ENTRY` macros, and to any other macro that assumes that its type parameter is derived from `IUnknown`.

[1] Types that are not standard layout are not required to place the first named base class at offset zero relative to the derived class, so this is something worth checking even if you think everything is set up correctly.

Raymond Chen

Follow