When MSDN says NULL, is it okay to use nullptr?

devblogs.microsoft.com/oldnewthing/20180307-00

March 7, 2018

Raymond Chen

In various places, MSDN will talk about the behavior corresponding to the case where a handle type has the value `NULL`. A customer wanted to know whether it was safe to use `nullptr` in such cases, or whether they have to use `NULL`.

Although the programming languages used by MSDN for documenting Windows are putatively C and C++, MSDN understands that a lot of people write code for Windows in other languages, and therefore it tries to avoid relying on language subtleties.

Esoteric definitions for the term `NULL` is one of those language subtleties.

Formally, the C and C++ languages permit the following definitions for the `NULL` macro:

`NULL`	`0`	`(void*)0`	`nullptr`
C	allowed	allowed	not allowed[1]
C++	allowed	not allowed[2]	allowed

If `NULL` is defined as `(void*)0` in C or as `nullptr` in C++, then it can be assigned only to a pointer type. And since MSDN cannot control how the C and C++ header files define `NULL`, it needs to work with any definition that is permitted by the corresponding standards. Which means that saying `NULL` implies that the underlying type is a pointer type.

Therefore, you are welcome to write `nullptr` instead of `NULL` if you're writing C++ code. You're also welcome to write anything else that produces a null pointer, such as

```
HMUMBLE h1 = HMUMBLE();
HMUMBLE h2 = HMUMBLE{};
HMUMBLE h3 = HMUMBLE(0);
HMUMBLE h4 = (HMUMBLE)0;
HMUMBLE h5 = 3 - 3;
```

But most people would probably prefer you to write `NULL` or `nullptr`.

As noted, MSDN understands that a significant portion of its readership is not fluent in the subtleties of C and C++. When it writes `NULL`, it means the obvious thing: A null pointer. You can translate that into the appropriate construction for the language you are using. For example, for C#, you can use `null`, or if you are operating in raw `IntPtr`s, you can use `IntPtr.Zero`.

Bonus chatter: When MSDN says `NULL`, is it okay to use `0`? Yes, but you probably don't want to. Using `0` as a null pointer constant is permitted by the C and C++ languages for backward compatbility reasons, but it's not considered modern style.

Bonus bonus chatter: I'm told that the Visual C++ folks occasionally entertain the possibility of changing the definition of `NULL` to `nullptr`, which is permitted by the standard. However, this ends up breaking a lot of code which assumed that `NULL` is an integral constant evaluating to zero. For example:

```
void foo(char* p)
{
  char c = NULL; // would not work if NULL were defined as nullptr
  *p = NULL;     // would not work if NULL were defined as nullptr
  ...
}
```

Although that code is technically already broken, it manages to work if `NULL` is defined as `0`, and updating the definition in the language header files would break existing (albeit poorly-written) code.

[1] C does not have the `nullptr` keyword.

[2] C++ does not allow `NULL` to be defined as `(void*)0` because C++ does not permit implicit conversion from `void*` to arbitrary `T*`.

```
int* p = (void*)0; // allowed in C but not C++
```

Raymond Chen

Follow