Use the #error directive to check whether the compiler even sees you

devblogs.microsoft.com/oldnewthing/20080409-00

April 9, 2008

Raymond Chen

You may find yourself in a twisty maze of `#ifdef`s. Or you may be wondering why your macros aren't working.

> I have these lines in my header file:
>
> ```
> #define MM_BUSY     0x0001
> #define MM_IDLE     0x0002
> ```
>
> but when I try to use them, I get errors.
>
> ```
> sample.cpp(23): error C2065: 'MM_BUSY': undeclared identifier
> sample.cpp(40): error C2065: 'MM_IDLE': undeclared identifier
> ```
>
> Any idea why this is happening?

First, make sure the compiler even sees you. Notice that for macros, generating a preprocessed file doesn't accomplish anything since `#define`s don't show up in the preprocessor output. (They are preprocessor *input*.) What I do is use the `#error` directive. Add it to the header file and recompile.

```
#define MM_BUSY     0x0001
#define MM_IDLE     0x0002
#error Did we get here?
```

If you get

```
sample.h(80) : error C1189: #error :  Did we get here?
```

then you know that the line is indeed being compiled and that somebody after you is doing an `#undef MM_BUSY`. If not, then you get to investigate why the lines in the header file are being ignored. For example, they might be hidden by an `#ifdef`, or (if you're using Visual Studio with precompiled headers), your `#include` directive might be ignored due to an overriding precompiled header directive. You can scatter `#error` directives into other parts of the header file (or other header files) to narrow down why your lines are being skipped.

Raymond Chen

Follow