The curse of the redefinition of the symbol HLOG

devblogs.microsoft.com/oldnewthing/20150724-00

July 24, 2015

Raymond Chen

A customer was running into a compiler error complaining about redefinition of the symbol `HLOG`.

```
#include <pdh.h>
#include <lm.h>

...
```

The result is

```
lmerrlog.h(80): error C2373: 'HLOG' redefinition; different type modifiers
pdh.h(70): See declaration of 'HLOG'
```

"Our project uses both performance counters (`pdh.h`) and networking (`lm.h`). What can we do to avoid this conflict?"

We've seen this before. The conflict arises from two problems.

First is hubris/lack of creativity. "My component does logging. I need a handle to a log. I will call it `HLOG` because (1) I can't think of a better name, and/or (2) obviously I'm the only person who does logging. (Anybody else who wants to do logging should just quit their job now because it's been done.)"

This wouldn't normally be a problem except that Win32 uses a global namespace. This is necessary for annoying reasons:

- Not all Win32 languages support namespaces.
- Even though C++ supports namespaces, different C++ implementations decorate differently, so there is no agreement on the external linkage. (Indeed, the decoration can change from one version of the C++ compiler to another!)

Fortunately, in the case of `HLOG`, the two teams noticed the collision and came to some sort of understanding. If you include them in the order

```
#include <lm.h>
#include <pdh.h>
```

then `pdh.h` detects that `lm.h` has already been included and avoids the conflicting definition.

```
#ifndef _LMHLOGDEFINED_
typedef PDH_HLOG     HLOG;
#endif
```

The PDH log is always accessible via the name `PDH_HLOG`. If `lm.h` was not also included, then the PDH log is also accessible under the name `HLOG`.

Sorry for the confusion.

Raymond Chen

Follow