If you wonder why a function can't be found, one thing to check is whether the function exists in the first place

devblogs.microsoft.com/oldnewthing/20150204-00

February 4, 2015

Raymond Chen

One of my colleagues was frustrated trying to get some code to build. “Is there something strange about linking variadic functions? Because I keep getting an unresolved external error for the function, but if I move the function definition to the declaration point, then everything works fine.”

```
// blahblah.h
... other declarations ...
void LogWidget(Widget* widget, const char* format, ...);
...
// widgetstuff.cpp
...
#include "blahblah.h"
...
// some code that calls LogWidget
void foo(Widget* widget)
{
 LogWidget(widget, "starting foo");
 ...
}
// and then near the end of the file
void LogWidget(Widget* widget, const char* format, ...)
{
    ... implementation ...
}
...
```

“With the above code, the linker complains that `LogWidget` cannot be found. But if I move the implementation of `LogWidget` to the top of the file, then everything builds fine.”

```
// widgetstuff.cpp
...
#include "blahblah.h"
...
// move the code up here
void LogWidget(Widget* widget, const char* format, ...)
{
    ... implementation ...
}
// some code that calls LogWidget
void foo(Widget* widget)
{
 LogWidget(widget, "starting foo");
 ...
}
...
```

"I tried putting an explicit calling convention in the declaration, I tried using `extern "C"`, nothing seems to help."

We looked at the resulting object file and observed that in the case where the error occurred, there was an external reference to `LogWidget` but no definition. I asked, "Is the definition of the function `#ifdef`'d out by mistake? You can use this technique to find out."

That was indeed the problem. The definition of the function was inside some sort of `#ifdef` that prevented it from being compiled.

Sometimes, the reason a function cannot be found is that it doesn't exist in the first place.

Raymond Chen

Follow