The extern "C" specifier disables C++ mangling, but that doesn't mean it disables mangling

devblogs.microsoft.com/oldnewthing/20120525-00

May 25, 2012

Raymond Chen

The MSDN documentation on dllexport contains the following enigmatic paragraph, or at least did at the time I wrote this article:

> **dllexport** of a C++ function will expose the function with C++ name mangling. If C++ name mangling is not desired, either use a .def file (EXPORTS keyword) or declare the function as extern “C”.

I’ve seen this sentence misinterpreted as follows:

> **dllexport** of a C++ function will expose the function with C++ name mangling. To disable name mangling either use a .def file (EXPORTS keyword) or declare the function as extern “C”.

This is an understandable misinterpretation, but it is still a misinterpretation.

The root cause of the misinterpretation is that the author of this documentation was wearing C++-colored glasses. In the author’s mind, there are only two interesting cases:

1. C++ name mangling, where all the cool people are, and
2. everything else, for all the lamers.

Here is a precise formulation of the paragraph:

> **dllexport** of a C++ function will expose the function with C++ name mangling. If C++ name mangling is not desired, either use a .def file (EXPORTS keyword), which will expose the name without mangling, or declare the function as extern “C”, which will expose the name with C mangling.

Here’s a version of the paragraph that tries to take away the C++-colored glasses.

> **dllexport** exposes the function as it is decorated by the compiler. For example, if the function is a C++ function, it will be exposed with C++ name mangling. If the function is a C function, or has been declared as `extern "C"`, it will be exposed with C name mangling. To expose the function under its unmangled name (or to expose it via an alternate name), use use a .def file (EXPORTS keyword).

Behind the scenes: To forestall nitpickers, I had to go back to my copy of the C++ standard to make sure I filled in the blank in “The `extern "C"` ___________” correctly. Officially, `extern "C"` is a *storage class specifier*.

Raymond Chen

Follow