# Why can't I __declspec(dllexport) a function from a static library?

devblogs.microsoft.com/oldnewthing/20140321-00

March 21, 2014

Raymond Chen

A customer was having trouble exporting a function with the `__declspec(dllexport)` declaration specified, but found that if the function was in a static library, no function was exported. Why is that?

Let's go back to the classical model for linking. Code is pulled from a LIB file only if the linker encounters a reference to it. If the linker encounters no reference to any symbols offered by an OBJ file in that LIB, then that OBJ file is not included in the final image. (Remember, we're talking about OBJ files inside LIBs; explicitly-provided OBJ files are always included in the image, under the classical model.)

Now consider a LIB which contains an OBJ file that has a function marked `__declspec(dllexport)`. Suppose that no symbol offered by that OBJ file is ever required by the final image. That means that the OBJ file is never added to the image. And that means that the linker does not see the `__declspec(dllexport)` qualifier on the function inside the OBJ file (since the OBJ file was never used), so the function doesn't get exported.

Let's look at it another way: `__declspec(dllexport)` does not influence the linking process. All it does is add a little sticker to the function that says, "For export." When the linker adds functions to an image, it makes note of the sticker and adds it to the list of functions that it needs to export. But if the linker never sees the function, then it never sees the sticker.

In order to export a function from a static library, you need to force a reference to the function from the image. One way is to add an OBJ to the image that contains a dummy function that calls the function you want to export. That dummy function will trigger the resolution of the symbol from the static library, at which point the linker will see the sticker.

Another way is to use the `/INCLUDE` directive to create an artificial reference to the function from the command line, but this gets you into the fragile world of having to know the various name decoration schemes for different architectures.

The best solution is to use an explicit `DEF` file, since that also gives you a chance to do other things like remove the decorations from the function (so that it can be `GetProcAddress`ed).

**Exercise**: "But sometimes the `__declspec(dllexport)` works from a static library, even though I did none of those special things." Explain.

Raymond Chen

**Follow**