If you return FALSE from DLL_PROCESS_ATTACH, will you get a DLL_PROCESS_DETACH?

devblogs.microsoft.com/oldnewthing/20080808-00

August 8, 2008

Raymond Chen

If you return `FALSE` from `DLL_PROCESS_ATTACH`, will you get a `DLL_PROCESS_DETACH`?

Yes.

No.

...

Yes.

All three answers are correct, for different formulations of the question.

From the kernel's point of view, the answer is a simple Yes. If a DLL's entry point returns `FALSE` to the `DLL_PROCESS_ATTACH` notification, it will receive a `DLL_PROCESS_DETACH` notification.

However, most C and C++ programs do not use the raw DLL entry point. Instead, they use the C runtime entry point, which will have a name something like `DllMainCRTStartup`. That entry point function does work to manage the C runtime library and calls your entry point (which you've probably called `DllMain`) to see what you think.

If you compiled your program prior to around 2002 and your `DllMain` function returns `FALSE` in response to the `DLL_PROCESS_ATTACH` notification, then the C runtime code says, "Oh, well, I guess I'm not running after all" and shuts itself down. When the kernel calls the C runtime entry point with the `DLL_PROCESS_DETACH` notification, the C runtime says, "Oh, I'm already shut down, thanks for asking" and returns immediately, which means that your entry point is not called with the `DLL_PROCESS_DETACH` notification. In other words, if you wrote your program prior to around 2002, the answer is No.

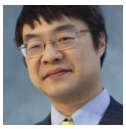
Sometime in 2002 or maybe 2003, the C runtime folks changed the behavior. If your `DllMain` function returns `FALSE` in response to the `DLL_PROCESS_ATTACH` notification, you will nevertheless get the `DLL_PROCESS_DETACH` notification. In other words, if you

wrote your program after around 2002 or maybe 2003, then the answer is Yes. Why change? Maybe they wanted to match the kernel behavior more closely, maybe they considered their previous behavior a bug. You'll have to ask them.

What does this mean for you, the programmer? Some people may look at this and conclude, "Well, now that I know how each of the specific scenarios works, I can rely on knowing the behavior that results from the scenario I'm in. For example, since I'm using Visual Studio 2008, the answer is Yes." But I think that's the wrong conclusion, because you usually do not have total control over how your program is compiled and linked. You may share your code with another project, and that other project may not know that you are relying on the behavior of a specific version of Visual Studio 2008; they will compile your program with Borland C++ version 5.5,[1] and now your program is subtly broken. My recommendation is to write your `DllMain` function so that it works correctly regardless of which scenario it ends up used in. (And since you shouldn't be doing much in your `DllMain` function anyway, this shouldn't be too much of a burden.)

Footnote

[1]I do not know what the behavior of Borland C++ version 5.5 is with respect to returning `FALSE` from `DllMain`. I didn't feel like doing the research to find a compiler whose behavior is different from Visual Studio 2008, so I just picked one at random. I have a 50/50 chance of being right.



Raymond Chen

Follow