When are global objects constructed and destructed by Visual C++?, redux

devblogs.microsoft.com/oldnewthing/20160930-00

September 30, 2016

Raymond Chen

Today we're going to make some clarifications to this table, which came from an earlier article:

When does it run?	**Constructor**	**Destructor**
Global object in EXE	C runtime startup code	C runtime DLL hired lackey
Global object in DLL	C runtime `DLL_PROCESS_ATTACH` prior to `DllMain`	C runtime `DLL_PROCESS_DETACH` after `DllMain` returns

It turns out that the upper right corner of the diagram actually splits into two cases. The table lists what happens if the process terminates by calling `ExitProcess`. The thing that makes termination with `ExitProcess` interesting is that the first (and only) time the C runtime library learns about it is when the C runtime library itself receives its `DLL_PROCESS_DETACH` notification, and we saw last time that by the time this notification arrives, it could very well already be too late.

The escape here is to exit the program not by calling `ExitProcess` but rather by calling the C runtime `exit` function. When you do that, the C runtime gets control (by virtue of the fact that you explicitly called it), so it can run down your executable's global objects right away, before calling the operating system's `ExitProcess` function. That way, the global objects are run down while all of the dependent DLLs are still in memory.

Let's update our table:

When does it run?	**Constructor**	**Destructor**

		Ends with exit()	Ends with ExitProcess()
Global object in EXE	C runtime startup code	Prior to ExitProcess	C runtime DLL hired lackey
Global object in DLL	C runtime `DLL_PROCESS_ATTACH` prior to `DllMain`	C runtime `DLL_PROCESS_DETACH` after `DllMain` returns	

The C and C++ language standards say nothing about what happens if you exit a process by calling some operating system low-level process termination function. Which makes sense, because the C and C++ language standards deal with the standard, not with operating system-specific stuff. I believe that more recent versions of the C runtime library take advantage of this and say, "You know what? If you exit the process by calling `ExitProcess`, then I'm simply not going to destruct anything. Serves you right for invoking behavior not covered by the standard." In those cases, the upper right corner changes from "C runtime DLL hired lackey" to "never".

Raymond Chen

Follow