The lackey catastrophe

devblogs.microsoft.com/oldnewthing/20160929-00

September 29, 2016

Raymond Chen

We encountered a real problem with global object destruction in Explorer. The object in question was an RAII container for a graphics object, so its destructor destroyed the graphics object. But the call to destroy the object was crashing.

```
GDI32!IS_INDEX_IN_USER_SHARED_ARRAY+0x17
GDI32!HANDLE_TO_INDEX+0x1a
GDI32!DeleteObject+0x2a
explorer!CBitmap::~CBitmap+0x20
msvcrt!doexit+0xb6
msvcrt!_cexit+0xb
msvcrt!__CRTDLL_INIT+0x9f
ntdll!LdrxCallInitRoutine+0x16
ntdll!LdrpCallInitRoutine+0x43
ntdll!LdrShutdownProcess+0x1c1
ntdll!RtlExitUserProcess+0x96
kernel32!ExitProcess+0x32
explorer!wWinMain+0x4ef
explorer!WinMainCRTStartup+0x151
KERNEL32!BaseThreadInitThunk+0x24
ntdll!__RtlUserThreadStart+0x2b
ntdll!_RtlUserThreadStart+0x1b
```

What's going on?

The call to `DeleteObject` was occurring after `GDI32` had already run its ~~`DLL_PROCESS_ATTACH`~~ `DLL_PROCESS_DETACH`. As a result, it was calling into a DLL that had already uninitialized, so bad things happen.

But wait, how can you call into a DLL that has already uninitialized? The EXE links to the DLL via a load-time dependency, so the EXE should uninitialize first. But it's not. Why not?

Recall how global objects are constructed and destructed. If the global object had been in a DLL, then indeed the loader dependency analysis would have seen that the global object's DLL depends upon `GDI32`.

As we saw in the earlier discussion, executables do not have `DLL_PROCESS_DETACH`. You can look at the situation in two ways: One interpretation is that the executable has already stopped running at the point it calls `ExitProcess`. All that's left is to shut down the DLLs. Another interpretation is that the executable is *always* running (seeing as DLLs run in the context of a process), so there's no point trying to wait until the executable has "finished" because if you did that, you'd be waiting forever.

Anyway, regardless of how you choose to look at the situation, the problem is the same: The lackey we hired to run down our global objects is running them down too late.

This is a case where the C runtime is doing the best it possibly can, but it's still not good enough.

Next time, we'll look at one possible extrication from this quandary.

Raymond Chen

Follow