If it's not yours, then don't mess with it without permission from the owner

devblogs.microsoft.com/oldnewthing/20100430-00

April 30, 2010

Raymond Chen

It's surprising how many principles of real life also apply to computer programming. For example, one of the rules of thumb for real life is that if something doesn't belong to you, then you shouldn't mess with it unless you have permission from the owner. If you want to ride Jimmy's bike, then you need to have Jimmy's permission. Even if Jimmy leaves his bicycle unlocked in his driveway, that doesn't mean that it's available for anyone to take or borrow. In computer programming, the code that creates an object (or on whose behalf the object is created) controls what is done with the object, and if you're not that component, then it's only right to get the permission of that component before you start messing with that it thought was its private property. Application compatibility is, in large part, dealing with programs which violate this rule of civilized society, programs which directly manipulate the contents of list views they did not create, use reflection to access private members of classes, that sort of thing. But I won't use that as the motivating example this time, because you're all sick and tired of that. Instead, let's look at the low-fragmentation heap. The question is, "Under what conditions can I convert a heap to a low-fragmentation heap?" Well, if you called `HeapCreate`, then that heap is yours and you decide what the rules are. If you want that heap to be a low-fragmentation heap, then more power to you. If you didn't call `HeapCreate` then that heap doesn't belong to you; you're just a guest. But of course the owner of the heap can grant permission to you, at which point you are free to do whatever it was the owner said you could do. If Jimmy says, "You can borrow my bike if it's just sitting in the driveway," then you can borrow his bicycle if it is just sitting in the driveway. But if it's in the garage, then you can't borrow it. And even if it's sitting in the driveway, you can't sell it. You can only borrow it. Okay, let's look at heaps again. If you are an executable, then the process heap was created on your behalf. (This is not obvious, but that's the guidance I've received from the people who work with this sort of thing.) Therefore, if you want, you can call `GetProcessHeap` and convert that heap to a low-fragmentation heap. It's the heap for your process, so if you want it to be a low-fragmentation heap, the heap folks say that's okay with them. On the other hand, if you're writing a DLL, then the process heap does *not* belong to you, nor was it created on your behalf. It belongs to the executable that loaded your DLL, and it is that executable which decides what type of heap it wants. If you would prefer that your DLL use a low-fragmentation heap, you can include that

in the guidance in your DLL's documentation, but be aware that the process heap is shared with all DLLs in the process, so the hosting application may not be able to comply with your guidance if it is also using another DLL whose guidance documentation says that it should *not* be used with a low-fragmentation heap. If a low-fragmentation heap is really important to your DLL, then you can create your own heap with `HeapCreate` and set it into low-fragmentation mode. When you create a heap with `HeapCreate`, it's your heap, and you get to decide what the rules are. If you use the C runtime library default heap, then that heap is under the control of the C runtime library, and you don't have the rights to change its parameters. However, the C runtime library is one of the examples where you're allowed to use an object that's not yours if you have permission from the owner: The `_get_heap_handle` function was specifically created so that you could convert the heap to a low-fragmentation heap. But now that you've unwrapped one layer of ownership, there is still the matter of which of the C runtime's clients is the decision-maker with regard to how that heap is to be configured? Remember that a DLL is a guest in the host process. You don't go changing the carpets in someone's house just because you're visiting.

If you linked the C runtime library statically, then you are the only client of that heap, and you are therefore free to convert it to a low-fragmentation heap. (If you bring your own towels to someone's house, then you are free to abuse them in any manner you choose.) On the other hand, if you linked the C runtime library dynamically, then you're using the shared C runtime heap, and the authority to determine the mode of that heap belongs to the hosting executable.

Raymond Chen

Follow