What's the difference between FreeResource and, say, DestroyAcceleratorTable

devblogs.microsoft.com/oldnewthing/20110307-00

March 7, 2011

Raymond Chen

MaxMax asks a number of resource-related questions, starting with "How do you Unlock a LockResource?" and culminating in "What are the differences between `FreeResource` and `DestroyAcceleratorTable`, `DestroyObject`, etc.? It would be much easier to use a single function instead of a collection of five." It helps if you understand the history of resources, because the functions were designed back when resources were managed very differently from how they are today. The usage pattern is still the same:

- `FindResource`
- `LoadResource`
- `LockResource`
- use the resource
- `UnlockResource`
- `FreeResource`

You unlock a resource by calling, um, `UnlockResource`. Although the usage pattern is the same, the mechanism under the covers is completely different. In 16-bit Windows, loading a resource entailed allocating a chunk of memory, then filling that memory block from the disk image. In Win32, resources are mapped into the address space as part of the image; there is no memory allocation and no explicit loading. The next thing to understand is that resources are just blobs of binary data. They are not live objects. It's not like there's a `HBITMAP` sitting in there just waiting to be found. Think of resource data as a set of blueprints. If you call `FindResource` + `LoadResource` + `LockResource`, you wind up with the blueprints for a radio, but that's not the same as actually having a radio. To do that, you need to hand the radio blueprints to somebody who knows how to read electronic schematic diagrams and who knows how to solder wires together in order to convert the potential radio into an actual radio. If you've been following the sporadic series on the format of resources, you'll know that these schematic diagrams can often be quite complicated. The `LoadBitmap` function first does the `FindResource` + `LoadResource` + `LockResource` dance to locate the bitmap blueprint, but then it needs to actually make the bitmap, which is done by parsing the raw resource data and trying to make sense of it, calling functions like `CreateBitmap` and `Set-`

`DIBits` to convert the blueprint into an actual bitmap. That's why, if you use these helper functions like `LoadAccelerators` to convert the blueprint into an object, you need to use the corresponding cleanup function like `DestroyAcceleratorTable` when you want to destroy the object. You have to use the correct cleanup function, of course. You can't destroy a bitmap with `DestroyAcceleratorTable` any more than you can put a radio in the clothing drop bin.

Just like when the radio guy returns the original blueprints plus a brand new radio, you return the blueprints to the library, but if you want to destroy the radio, you have to take it to the electronics recycling facility.

Raymond Chen

Follow