Destroying the module and resource information associated with an icon

August 22, 2008

Raymond Chen

We've seen that icons and cursors know where they came from, and the window manager uses this information when you ask it to change the size of an icon. But not all icons carry this information, only icons created by passing a `HINSTANCE` and a resource name.

You can use this to your advantage if you want to destroy the module and resource information associated with an icon. For example, the `CreateIconIndirect` function creates an icon from raw bitmap information without reference to an `HINSTANCE` or a resource name. This allows you to create icons at runtime, but it also allows you to create an icon that "throws away" the bonus information.

```
HICON CopyIconWithoutResourceInfo(HICON hicoSrc)
{
  ICONINFO ii;
  HICON hico = NULL;
  if (GetIconInfo(hicoSrc, &ii)) {
    hico = CreateIconIndirect(&ii);
    if (ii.hbmMask) DeleteObject(ii.hbmMask);
    if (ii.hbmColor) DeleteObject(ii.hbmColor);
  }
  return hico;
}
```

Now, throwing away this information is a desperation move, because it prevents the window manager from using the original resource information when resizing an icon, resulting in ugly stretched icons.

You might even be throwing this information away by mistake. For example, if your program is asked to produce an icon, it's best if you load the icon with a function like `LoadImage` because that records the bonus information; if the caller decides to resize the icon, it can do so with the `CopyImage` function while retaining full fidelity. On the other hand, if you use a function like `ExtractIcon` or `CreateIconFromResource`, that will not have the bonus information, and any icon stretching that takes place will end up looking pretty ugly.

[Raymond is currently away; this message was pre-recorded.]

Raymond Chen

Follow