If the RegisterClass function takes ownership of the custom background brush, why is it leaking?

devblogs.microsoft.com/oldnewthing/20231218-00

December 18, 2023

Raymond Chen

A customer found that their program was leaking GDI resources, but they couldn't figure out why.

They suspect that it may have something to do with a recent change they made. They discovered in the fine print of the documentation of the `hbrBackground` member of the `WNDCLASS` structure

> The system automatically deletes class background brushes when the class is unregistered by using **UnregisterClass**. An application should not delete these brushes.

The code responsible for registering classes was registering the classes with brushes it did not own. When the original owner destroyed the brushes, that created a double-free bug. To fix the problem, they had their class registration code create a duplicate of the brush and register with the duplicate.

This fixed the double-free bug, but somehow it introduced a resource leak. Were they incorrect to duplicate the brush? Are all these duplicates being leaked? It seemed so, because most of the time, the program ran out of resources on the `CreateBrushIndirect` call in their brush duplication function, which suggests that these brushes are being leaked after all.

What's going on?

After being reassured that the `RegisterClass` function does indeed take ownership of the brushes used for class registrations, they went and studied their code more closely and found the problem.

The issue was that they created a brush for the class registration, expecting the `Register-Class` to take responsibility for destroying the brush when it's no longer needed. But they neglected to deal with the case where they register the same class more than once.

The code registers classes on demand, and before they create a window, they call `RegisterClass` to register the class just in time. If the class is already registered, then `RegisterClass` fails with `ERROR_CLASS_ALREADY_EXISTS`, in which case `RegisterClass` does *not* take responsibility for the `hbrBackground` brush. In the case that `RegisterClass` fails, the code needs to delete the brush manually.

The way to write the code will vary depending on which RAII class library you're using (if you're using one at all). If you're using the Windows Implementation Library, then it would go something like this:

```
/* Asssume DuplicateBrush makes a copy of a brush */
wil::unique_hbrush backgroundBrush = DuplicateBrush(hbrBackground);
RETURN_LAST_ERROR_IF(!backgroundBrush);

WNDCLASS wc{};
wc.hbrBackground = backgroundBrush.get();
⟦ other WNDCLASS initialization goes here ⟧

if (RegisterClass(&wc)) {
    // System owns the brush now.
    backgroundBrush.release();
}
```