The FILE_FLAG_DELETE_ON_CLOSE flag applies to the handle, also known as the file object, which is not the same as the file

devblogs.microsoft.com/oldnewthing/20160108-00

January 8, 2016

Raymond Chen

A customer was having trouble with the `FILE_FLAG_DELETE_ON_CLOSE` flag. "We create a file that we want to be deleted automatically when we're finished with it. Whenever we open the file, we ask for `GENERIC_READ | GENERIC_WRITE`, and we allow all sharing (including `FILE_SHARE_DELETE`), and we pass `FILE_FLAG_DELETE_ON_CLOSE`. We can open the file multiple times in this manner, but as soon as we close any of the handles, we can't open any new ones."

Yup, that's right.

The `FILE_FLAG_DELETE_ON_CLOSE` flag means that the file will be deleted when the last handle to the file object is closed. This is not the same as closing the last handle to the *file*, however. Each call to `CreateFile` creates a new file object. You can create multiple handles to the same file object by calling `DuplicateHandle`. (We saw this when we discussed the effect of handle duplication on the file pointer.)

When the last handle to a file object is closed, the file object deletes the underlying file. Existing file objects will still refer to the file, but no new ones are allowed. When there are no more file objects, then the directory entry is removed. (Removing the directory entry is what most people think of when they talk about "deleting a file", but it's actually just the last step in the process.)

Going back to the customer's problem: It looks like they really want the file to remain valid (including allowing further `CreateFile` calls to succeed) for as long as any open handle continues to refer to the file. Fortunately, the customer needed the handle only to create a memory-mapped view. The file pointer was not important. Therefore, the customer could use `DuplicateHandle` instead of `CreateFile` to get additional handles to the file. Since all of the handles refer to the same file object, the file object will not delete the file until all of the handles are closed.

Raymond Chen

Follow