If I duplicate a handle, can I keep using the duplicate after closing the original?

devblogs.microsoft.com/oldnewthing/20140711-00

July 11, 2014

Raymond Chen

A customer asked whether it was okay to use a duplicated handle even after the original handle was closed. Yes. That's sort of why you would duplicate it. Duplicating a handle creates a second handle which refers to the same underlying object as the original. Once that's done, the two handles are completely equivalent. There's no way to know which was the original and which is the duplicate. Either handle can be used to access the underlying object, and the underlying object is not torn down until all handles to it have been closed. One tricky bit here is that since you have two ways to refer to the same thing, changes made to the object via one handle will be reflected when observed through the other handle. That's because the changes you're making are to the object itself, not to the handle. For example, if you duplicate the handle to an event, then you can set the event via either handle. That may all sound obvious, but one thing to watch out for is the case of file handles: The current file position is a property of the file object, not the handle. Say you duplicate a file handle and give the original to one component and the duplicate to another. Now, when either component reads from or writes to the file, it's going to change the current position of the file object, and consequently may confuse the other component (who may not have expected the current position to be changing). Also, if the underlying file is a synchronous file handle, the file operations on the underlying file will be synchronized. If one component starts a read, the other component won't be able to access the file object until that read completes. If you want to create a second handle to a file that has its own file pointer and is not synchronized against the first file handle, you can use the `ReOpenFile` function to create a second file object with its own synchronization and its own file position, but which refers to the same underlying file.

(Don't forget to get your sharing modes right! The second file object's access and sharing modes must be compatible with access and sharing modes of the original file object. Otherwise the call will fail with a sharing violation.)

Raymond Chen

Follow