ReadDirectoryChangesW reads directory changes, but what if the directory doesn't change?

devblogs.microsoft.com/oldnewthing/20110812-00

August 12, 2011

Raymond Chen

The `ReadDirectoryChangesW` function reads changes to a directory. But not all changes that happen to files in a directory result in changes to the directory. Now, it so happens that nearly all changes to a file in a directory really do result in something happening to the file's directory entry. If you write to a file, the last-write time in the directory entry changes. If you rename a file, the name in the directory entry changes. If you create a file, a new directory entry is created. But there are some changes that do not affect the directory entry. I've heard rumors that if you write to a file via a memory-mapped view, that will not update the last-write time in the directory entry. (I don't know if it's true, but if it's not, then just pick some other file-modifying operation that doesn't affect the directory entry, like modifying the contents of a file through a hard link in another directory, or explicitly suppressing file timestamp changes by calling `SetFileTime` with a timestamp of `0xFFFFFFFF`FFFFFFFF`.) The point is that since these changes have no effect on the directory, they are not recognized by `ReadDirectoryChangesW`. The `ReadDirectoryChangesW` function tells you about changes to the directory; if something happens that doesn't change the directory, then `ReadDirectoryChangesW` will just shrug its shoulders and say, "Hey, not my job." If you need to track all changes, even those which do not result in changes to the directory, you need to look at other techniques like the change journal (a.k.a. USN journal). The intended purpose of the `ReadDirectoryChangesW` function is to assist programs like Windows Explorer which display the contents of a directory. If something happens that results in a change to the directory listing, then it is reported by `ReadDirectoryChangesW`. In other words, `ReadDirectoryChangesW` tells you when the result of a `FindFirstFile` / `Find-NextFile` loop changes. The intended usage pattern is doing a `FindFirstFile` / `Find-NextFile` to collect all the directory entries, and then using the results from `Read-DirectoryChangesW` to update that collection incrementally. In other words, `Read-DirectoryChangesW` allows you to optimize a directory-viewing tool so it doesn't have to do full enumerations all the time.

This design philosophy also explains why, if too many changes have taken place in the directory between calls to `ReadDirectoryChangesW`, the function will fail with an error called `ERROR_NOTIFY_ENUM_DIR`. It's telling you, "Whoa, like so much happened that I couldn't keep track of it all, so you'll just have to go back and do another `FindFirstFile` / `FindNextFile` loop."

Raymond Chen

Follow