How can I recognize whether two handles refer to the same underlying file?

devblogs.microsoft.com/oldnewthing/20220128-00

January 28, 2022

Raymond Chen

Last time, we learned how to recognize file systems that don't support 64-bit unique file identifiers. But what do you do want to detect whether two handles refer to the same underlying file, and the file system won't give you 64-bit unique file identifiers?

The first step is to compare the volume serial numbers. If they are different, then stop. The files belong to different volumes.

If the volume serial numbers match, then use the `GetFileInformationByHandleEx` function and ask for the `FileIdInfo`. This will give you the 128-bit file identifier. If you can get the 128-bit file identifier, then compare them. (NTFS uses 64-bit file identifiers, but it will return them as a 128-bit value if you ask for one.)

If the volume doesn't support 128-bit file identifiers, then maybe it's an older file system that doesn't understand the "Please give me the 128-bit file identifier" function, so try again with the `GetFileInformationByHandle` and compare the 64-bit file identifiers, provided they aren't zero or ``0xFFFFFFFF`FFFFFFFF``.

If the volume doesn't support 64-bit identifiers either, then the situation is getting kind of desperate. My colleague Malcolm Smith suggests calling `GetFinalPathNameByHandle` with `FILE_NAME_NORMALIZED` to try to get the file names into some sort of sane state, and then comparing the names.

This last step isn't foolproof, because it will be faked out by hard links, but it's probably the best you can do under the circumstances. Let's hope that the remote file systems which don't support unique identifiers don't support hard links either.

Bonus chatter: The SMB protocol (also known as CIFS) can get into a situation where a single share maps to multiple file systems, which means that the unique identifier is itself just an approximation. If two underlying file systems both use the same unique identifier for different files, then the unique identifier sent over the network via SMB will not actually be unique. But it's trying the best it can under the circumstances.[1]

[1] This also explains why you might not see the `FILE_SUPPORTS_OPEN_BY_FILE_ID` flag on remote volumes. If the underlying volume is multiplexing multiple underlying file systems, then it won't be able to redeem those file IDs for files, since it doesn't know which file system each ID came from. (I guess it could just try them all, but that would defeat the purpose of striping the storage.)

Raymond Chen

Follow