A file can go by multiple names, but two files can't have the same name

devblogs.microsoft.com/oldnewthing/20081208-00

December 8, 2008

Raymond Chen

Thanks to short file names and hard links, a single file can go by multiple names. (And for the purpose of today's discussion, I'm treating the full path as the name instead of just the part after the last backslash. Don't make me bring back the nitpicker's corner.) For example, C:\PROGRA~1 and C:\Program Files are two possible names for the same directory thanks to short names. [Typo fixed 7:15am.] If you've created hard links, then you can give a single file two entirely unrelated names, and those names need not even be in the same directory.

On the other hand, you can't have two files with the same name. What would that even mean? Which one would you get if you issue an Open call? How would you open the other one? Heck, even before we get to this point: How do you even create two files with the same name? If you call `CreateFile` to create the "second" file, it'll just open the existing one!

That's why I am baffled by this question that asks:

> I've seen a few cases where people write their own version of GetLongPathName (usually because they need to support NT4) using FindFirstFile. Are there situations where that approach would return an incorrect path? Is it safe in practice because FindFirstFile("foo.bar") always will return the exact match first before returning "foo.barbaz"?

This question assumes that it's possible to have two files in a directory with the same name: One is the file "foo.bar"; the other is the file that goes by the long name "foo.barbaz" and the short name "foo.bar".

I'm not sure what the sequence of events would be that could result in two files with the same name. Here's one scenario:

1. Create file "foo.barbaz". It gets assigned the short name "foo.bar".
2. Create file "foo.bar". This creates a new file called "foo.bar" which conflicts with the short name of the existing file "foo.barbaz".

Except that's not what'll happen. When you perform step 2, the attempt to create the file "foo.bar" merely overwrites the existing file which has "foo.bar" as one of its names. Result: A single file with long name "foo.barbaz" and short name "foo.bar".

Another scenario might go like this:

1. Create file "foo.bar".
2. Create file "foo.barbaz". The file system auto-assigns the short name "foo.bar".

Except that's not what happens either. At the second step, the file system generates a short name like "foo~1.bar" specifically to avoid the name collision.

You can run these experiments yourself from the command line to confirm. The "dir /x" command will come in handy.

Raymond Chen

Follow