The Old New Thing

Why does FindFirstFile find short names?

20 Jul 2005 10:00 AM 21

The `FindFirstFile` function matches both the short and long names. This can produce somewhat surprising results. For example, if you ask for "*.htm", this also gives you the file "x.html" since its short name is "X~1.HTM".

Why does it bother matching short names? Shouldn't it match only long names? After all, only old 16-bit programs use short names.

But that's the problem: 16-bit programs use short names.

Through a process known as generic thunks, a 16-bit program can load a 32-bit DLL and call into it. Windows 95 and the Windows 16-bit emulation layer in Windows NT rely heavily on generic thunks so that they don't have to write two versions of everything. Instead, the 16-bit version just thunks up to the 32-bit version.

Note, however, that this would mean that 32-bit DLLs would see two different views of the file system, depending on whether they are hosted from a 16-bit process or a 32-bit process.

"Then make the `FindFirstFile` function check to see who its caller is and change its behavior accordingly," doesn't fly because you can't trust the return address.

Even if this problem were solved, you would still have the problem of 16/32 interop across the process boundary.

For example, suppose a 16-bit program calls `WinExec("notepad X~1.HTM")`. The 32-bit Notepad program had better open the file X~1.HTM even though it's a short name. What's more, a common way to get properties of a file such as its last access time is to call `FindFirstFile` with the file name, since the `WIN32_FIND_DATA` structure returns that information as part of the find data. (Note: `GetFileAttributesEx` is a better choice, but that function is comparatively new.) If the `FindFirstFile` function did not work for short file names, then the above trick would fail for short names passed across the 16/32 boundary.

As another example, suppose the DLL saves the file name in a location external to the process, say a configuration file, the registry, or a shared memory block. If a 16-bit program program calls into this DLL, it would pass short names, whereas if a 32-bit program calls into the DLL, it would pass long names. If the file system functions returned only long names for 32-bit programs, then the copy of the DLL running in a 32-bit program would not be able to read the data written by the DLL running in a 16-bit program.

Blog - Comment List MSDN TechNet

Comments

anonymous coward
20 Jul 2005 12:00 PM
#

I got bitten badly by this once. I used some tool that made HTML files for me. On perusing the directory listing, I saw that it had made files with an extension of .htm.

Fortunately it had the option of using the correct extension and I got it to make them with an extension of .html. Now my directory had both .htm and .html files. I thought I'd quickly delete the redundant *.htm files and had all of my data wiped clean.

Although the backwards compatibility was appreciated in 1994, I do sometimes like to scream in the general direction of Microsoft that it is no longer necessary for me and hasn't been for many years, and is especially unappreciated when I lose data in 2005!

I guess it is time to go off and see if I can find the registry key that turns off short file name generation on NTFS and then check to see if Windows synthesizes the short names if the filesystem doesn't actually have them ...

GregM
20 Jul 2005 12:29 PM
#
My question would not be so much "Why does FindFirstFile find short names", but "Why is there no function like FindFirstFile that can be told to only find long names". I had to write my own FindFirstFile wrapper function that checked that the extension exactly matched what I provided, because *.mod would match foo.model and bar.model. If I then did a find on *.model, I would get foo.model and bar.model again.

asdf
20 Jul 2005 1:31 PM
#
On 64bit windows, does this still return the short file names?

Ben Cooke
20 Jul 2005 1:34 PM
#
Anonymous Coward,
Note that you can disable short filenames on your system if you're willing to suffer the consequences. Once you've done this, there will be no longfi~1.htm generated anymore and so there'll be nothing there to match *.htm.

In the process, though, you'll almost certainly end up breaking a few applications. That's what you get when you shun back-compat! :)

Ben Cooke
20 Jul 2005 1:35 PM
#
Anonymous Coward,
I didn't see your last paragraph the first five or six times I read your comment. Apologies for the redundancy.

vince
20 Jul 2005 3:17 PM
#

Ben Cooke said: > In the process, though, you'll almost certainly end up breaking a few > applications. That's what you get when you shun back-compat! :)

Or, more accurately, that's what you get when the backwards-compatibility is poorly designed.
I am sure whoever designed this had their reasons, but it is almost hard to picture a more obscure and confusing way to implement this.

It sometimes feels like that despite all the money MS has, and all the reputedly expert programmers, that much of the code seems to be written by confused summer interns.

Edge
20 Jul 2005 3:47 PM
#
Why not add a flag that can be used with FindFirstFileEx (passed in the dwAdditionalFlags parameter?) that can turn off this backwards compatibility? Surely anything done by the API would be faster than the wrappers GregM (above) and others have no doubt had to write to get around this.

In any event, thanks for the explanation. As someone who also also been tripped up by this, it's always interesting to hear why it is the way it is.

foxyshadis
20 Jul 2005 3:53 PM
#
Windows won't synthesize the names. The biggest problem I see is that 32-bit packages came in 16-bit installers as late as 2001, although they were mostly gone by 1999. Nowadays I recognize the symptoms and can relocate them to a folder without spaces or long names, but it sure was frustrating when I didn't know why it broke.

Hopefully in longhorn this option will be off by default in home installations. For business edition, well, who knows.

A
20 Jul 2005 4:35 PM
#
If only Win95TruncatedExtensions defaulted to 0...

http://www.microsoft.com/resources/documentation/Windows/2000/server/reskit/en-us/regentry/28234.asp

Andreas Häber
20 Jul 2005 9:39 PM
#
Anon Coward: On Windows XP (and I assume Windows 2003) just type this command: "fsutil behavior query disable8dot3" to check the current short name generation setting and "fsutil behavior set disable8dot3 1" to disable short name generation.

Norman Diamond
20 Jul 2005 9:53 PM
#
> After all, only old 16-bit programs use
> short names.

One time in the registry of a Windows XP machine I noticed several string values that were pathnames in the 8.3 format (...\PROGRA~1\... or was it ...\DOCUME~1\...). I haven't hunted for them but I'm pretty sure they were associated with 32-bit programs not 16-bit programs. Sure that's in the registry not on disk, but if I removed 8.3 filenames from the disk then surely those programs would fail.

By the way when 8.3 filenames are disabled in an NTFS partition, does that delete existing short names or only disable generation of new ones?

josh
21 Jul 2005 12:12 AM
#
"Hopefully in longhorn this option will be off by default in home installations."

Or XP x64, which doesn't support 16-bit processes...

Matthew W. Jackson
21 Jul 2005 12:30 AM
#
I second the idea about a flag to disable this behavior. Surely no 16-bit software would use a flag that didn't exist until now.

I also agree that disabling short filename generation at some point would be the way to go, although I've seen a few too many 32-bit programs that cannot handle spaces in their file names (one major product by a major database vendor that is not Microsoft comes to mind), so I resort to setting them up in PROGRA~1. I've seen other 32-bit programs store short filenames themselves, even when they're capable of using long filenames.

foxyshadis
21 Jul 2005 1:21 AM
#
Only disables new ones.

I felt kind of bad recently, when I found out apache+php won't shell out to a path with spaces, that my workaround was using progra~1. I should file a bug report, but I'm not sure with which.

James
21 Jul 2005 3:32 AM
#
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"?

Or does this wacky behavior apply only when wildcards are used? I don't imagine that many 16-bit apps called FindFirstFile with an exact path as an argument, so backward compatibility there probably wouldn't be a concern.

Mike
21 Jul 2005 6:33 AM
#
Raymond, I know this is the wrong place for requests, but seeing you mention generic thunks made me wonder why the same approach isn't being used for 64 bit Windows. It seems that on WOW64, only syscalls are thunked - every other DLL an app uses except NTDLL has to have a 64-bit pair. I know that's a lot simpler than using thunks, so is it just that these days, shipping/loading two copies of every DLL isn't as costly as it once was?

Martin Plante
21 Jul 2005 9:35 AM
#
Would have it been possible to only support short filenames in FindFirstFileA? I understand it would have meant duplication of a lot of stuff. Are there other reasons FindFirstFileW required the same behavior?

Just curious.

Norman Diamond
21 Jul 2005 9:33 PM
#
Thursday, July 21, 2005 12:30 AM by Matthew W. Jackson
> I've seen a few too many 32-bit programs
> that cannot handle spaces in their file
> names (one major product by a major database
> vendor that is not Microsoft comes to mind),

And one major DDK by a major DDK vendor.

Thursday, July 21, 2005 9:35 AM by Martin Plante
> Would have it been possible to only support
> short filenames in FindFirstFileA?

That would be disastrous. FindFirstFileA is not capable of reporting every match that it finds. If you need reports of all matching existing filenames then you must call FindFirstFileW. It doesn't matter if your argument is a short name or a long name, you must call the W version.

Martijn

23 Jul 2005 6:05 AM
#

"If the file system functions returned only long names for 32-bit programs, then the copy of the DLL running in a 32-bit program would not be able to read the data written by the DLL running in a 16-bit program."

This is true for the reverse anyway (16-bit programs trying to read find files with their long file names), so maybe this should not be considered a problem.

余啊雷
30 Aug 2006 10:28 PM
#

PingBack from http://smallcode.weblogs.us/2006/08/31/dark-corners-in-microsoft-documentation/

余啊雷
8 Dec 2008 10:12 AM
#

PingBack from http://blogs.msdn.com/oldnewthing/archive/2008/12/08/9182990.aspx