How do I programmatically access the target path of a windows symbolic link?

Asked 16 years, 2 months ago Modified 2 years ago Viewed 11k times

14

Windows 6 (Vista and Server 2008) support proper symbolic links, which can be created via the CreateSymbolicLink function. But there doesn't appear to be a corresponding function for interrogating a symbolic link to obtain the path of the link's target.

I have found out that symbolic links are an implementation of reparse points, and so the reparse point functions can be used to obtain the target path. But the header files that I need to use reparse points appear to come with the Windows Driver Kit. Setting up this kit with VS2008 appears to be a non trivial task.

Is there a nice simple function that I've missed for obtaining a link's target, or do I really have to set up a windows driver development environment just to write code to access this information?

EDIT: Adam Mitz came up with the suggestion of GetFinalPathNameByHandle. This function works just great for local symlinks, but doesn't appear to work for resolving remote links (via a UNC path).

EDIT 2: At Adam's request, here are more details of what I've tried:

I initially went down the `FSCTL_GET_REPARSE_POINT` / `DeviceIoControl` route, but that yields a `REPARSE_DATA_BUFFER` structure. The headers that define this structure seem to exist solely within the Windows Driver Kit.

`GetFinalPathNameByHandle()` works fine when the link exists on a local disk (`C:\...\link` etc). Curiously, I found that I could obtain the handle to the link - and thus get the target - using `CreateFileW()` whether the `FILE_FLAG_OPEN_REPARSE_POINT` flag was specified or not, regardless of whether the target file exists.

When `CreateFileW()` and `GetFinalPathNameByHandle()` are used to interrogate a remote link though (`\\?\UNC\....`), things start to unravel. If `FILE_FLAG_OPEN_REPARSE_POINT` is specified, `GetFinalPathNameByHandle()` always returns the link path, not the target path. If `FILE_FLAG_OPEN_REPARSE_POINT` is not specified, then the target path is returned, but only if the target exists and is on the same machine as the link. If the link points to another machine, I get a network permissions error. If the link points to a local - non-existent - file, I get a file not found error.

windows **visual-studio-2008** **symlink**

Share

Improve this question

Follow

edited Nov 2, 2008 at 0:38

user3850

asked Oct 21, 2008 at 10:38

David Arno

43.2k ● 16 ● 88 ● 132

Please clarify if the symlink itself is on a remote server (via UNC) or if it's the symlink target that's a UNC path.
– Adam Mitz Oct 24, 2008 at 2:54

Also, I don't think you need DDK to read reparse points (no such thing as "parse points"). See the FILE_FLAG_OPEN_REPARSE_POINT flag on CreateFile or the FSCTL_GET_REPARSE_POINT flag for DeviceIoControl. Whether or not these will help depends on your answer to my earlier comment. – Adam Mitz Oct 24, 2008 at 3:23

Explorer does it somehow, I doubt it is directly by sending IO_CTL's. Try it: create a directory symbolic link, properties it and you'll get a "shortcut" tab with a grayed out destination.
– Fowl Mar 1, 2009 at 10:41

2 Answers

Sorted by: Highest score (default)

12

GetFinalPathNameByHandle

> A final path is the path that is returned when a path is fully resolved. For example, for a symbolic link named "C:\tmp\mydir" that points to

> "D:\yourdir", the final filesystem path would be "D:\yourdir".

Share Improve this answer Follow

answered Oct 21, 2008 at 11:36

Adam Mitz
6,033 ● 1 ● 30 ● 28

@Adam, unfortunately, whilst GetFinalPathNameByHandle() works for a local symlink, it returns the link, not the target for remote UNC-based symlinks. So I've "re-opened" the question. – David Arno Oct 23, 2008 at 15:48

I think I answered the question as asked (same as with your JNI/Mock question BTW... ahem). Maybe this is a new question. – Adam Mitz Oct 24, 2008 at 0:00

GetFinalPathNameByHandle is basically the answer, if one only wants to use local links. It looks like there is a bug in the function though as far as remote links are concerned. So I've marked this as the correct answer once more, and documented the caveat. – David Arno Oct 24, 2008 at 9:02

Unfortunately this function is not supported on win Xp – user13947194 Feb 23, 2022 at 13:34

This answer is not correct. If the handle refers to a file with multiple paths (hardlinks), only one of those paths will ever be returned no matter which one the symlink actually points to. The result is getting an entirely different path than what is actually in the symlink. Additionally, symlinks can be both relative or absolute, and that function only gives absolute answers. – LB-- Dec 20, 2022 at 19:12

1

Symbolic links can be both absolute or relative, and if they refer to a file which has multiple paths (hardlinks), the `GetFinalPathNameByHandle` function mentioned in the other answer will generally give you the wrong answer. If you open the symlink itself, you only get the path of the symlink itself, and if you open the path pointed to by the symlink, you will only ever get one of the paths of that file even if the symlink points to a different hardlink of it. If the file doesn't exist, you can't even use that function at all.

The correct answer appears to be to use C++17's `std::filesystem::read_symlink` function, this will actually read the reparse point data and give you the exact path the symlink contains, whether absolute or relative, and even works if the destination path does not exist. Additionally, it works in UWP applications too, despite the underlying Windows APIs apparently not being supported in UWP apps.

Share Improve this answer Follow

answered Dec 20, 2022 at 19:38

LB--
2,888 ● 2 ● 39 ● 79