The Old New Thing

Why doesn't Explorer have an interface for creating hard links?

28 Sep 2009 10:00 AM | 119

Nick asks why Explorer doesn't have UI for creating hard links.

Heck, while you're at it, why not ask "Why doesn't Explorer have a UI for hex-editing a file?"

Remember, all features start out with minus 100 points. Explorer is not under any obligation to expose every last NTFS feature, and it's certainly not going to help to expose an NTFS feature that even technical people don't understand. Just look at all the people who ask questions like "How can I tell if a file name is a hard link instead of a normal file?" or online explanations of hard links which claim that "A hard link is only a reference to the original file, not a copy of the file. If the original file is deleted, the information will be lost." I mean, if even a techno-geek doesn't understand hard links, how is your average user supposed to have a chance?

First, let's see how you would explain a hard link to an end user.

> When you create a hard link, you give the item an additional name. The file can be accessed by either its old name or its new name; they are equivalent. A file is not deleted until all its names are deleted.
>
> If you open a file by one name, make changes, and then save the file, the update is made to the file, and accessing the file by a second name will show the updated file, because the data for a file is separate from its name.
>
> Actually, that paragraph above is only true if the application which saves the file does so by writing directly to the name you used when you opened the file. If the application updates the file by creating a temporary file, then renaming the original to a backup name and renaming the temporary file, then updating a file by one name will sever its connection with other names for the file. The other names will continue to refer to the unmodified file; the name you used for the update will refer to the updated file. Different applications handle this situation differently; consult your application documentation for information on how it treats hard links.
>
> Note also that some backup programs have difficulty with hard links. Consult the documentation for your backup program for information on how it behaves when faced with hard links.
>
> If a file has multiple names, editing the file by using different names simultaneously may result in data corruption.
>
> If a file has multiple names, all the names must exist on the same drive.

Most users clicked Cancel at about the time you said, "When you create a hard link..." You'll probably lose the rest of them while you are busy warning about the potential problems with hard links, and the brave few who managed to stick it out to the end will be completely confused and hit Cancel. (The ones who hit Cancel early on were the smart ones.)

The problem with backup programs is particularly nasty. A backup program that is not hard-link-aware will miscalculate the necessary size of the backup volume, and when it attempts to restore the files, it will not restore the hard links.

There was a Windows XP PowerToy for creating hard links, but the backup team recommended that it not be released because so many third party backup programs exploded when they stumbled across files that had multiple names in the file system.

Blog - Comment List MSDN TechNet

Comments

Alexandre Grigoriev
28 Sep 2009 10:13 AM
#

Raymond,

Can you find out why Vista/2008/7 have those symbolic or hard links with legacy directory names? Is it to support legacy applications that use hardcoded paths? They don't work for that purpose anyway - "Access denied".

[*dup*. *-Raymond*]

John
28 Sep 2009 10:28 AM
#

Does Windows actually use hard links anywhere? I know Vista introduced junction points for backward compatibility purposes, but I am not aware of any applications that use hard links; it just seems like an extremely niche feature to support, especially when 99% of people (including myself) will get it wrong.

John Topley
28 Sep 2009 10:29 AM
#

It would seem that the name "hard" link is apposite, as hard links are indeed hard.

John Topley
28 Sep 2009 10:36 AM
#

@John: On a point of historical pendantry, I believe NTFS Junction Points were introduced with Windows 2000, not Windows Vista.

Christian
28 Sep 2009 10:39 AM
#

Would it be a such unreasonable obligation for explorer to implement e.g. a delete function that can handle EVERY feature NTFS exposes?

e.g. files with . in the name or \?

The posix layer allows such names.

Explorer should really be able to deal with all NTFS gotchas

[*"Why does Explorer show me a file where all operations fail except delete?" - Raymond*]

someone else
28 Sep 2009 10:40 AM
#

@John:

From what LinkShellExtension tells me, there's a metric crapload of hardlinks in system32 (Server 2008 R2). Most, if not all, of these are linked into WinSxS.

Does Explorer support junctions these days without extensions? Or does it still recursively delete the contents of a junction target instead of just the junction itself?

Pierre B.
28 Sep 2009 10:43 AM
#

John: I'm pretty sure Windows uses hard-link for some protected files and installed files, like drivers and such and for restoration point too, I believe. That's how it can have "backups" without using too much disk space.

BOFH
28 Sep 2009 10:43 AM
#

This "hard link" stuff that fsutil.exe does, is it separate from what's called junction points in the Windows Server 2000 Resource Kit documentation?

There was a tool in the Win2K RK called linkd.exe, and now there's also Sysinternals junction.exe that both create and delete junction points.

Junction points can be created to different volumes, which should mean that they are soft links, right?

Besides, junction points are only relevant to folders, so your warning regarding the "create temporary file/delete original/rename temporary" process does not apply there, I hope.

Karellen
28 Sep 2009 10:44 AM
#

What definition of "techno-geek" is in use here?

Most techno-geeks I know are of the Unix-variant-using variety, and I think all of them understand hard links, and don't think of them as that difficult a concept to grasp.

Why do Windows techno-geeks have problems here?

Or do they? If you're going to include any "self-proclaimed expert" on the internet who is willing to give advice on a subject, no matter how stupid, misinformed or flat-out wrong they actually are, as a member of an "expert" group, you're probably going to end up with a mighty poor opinion of experts in almost any field. Windows techno-geekery included.

John
28 Sep 2009 10:49 AM
#

@John Topley: I know junction points as a feature were introduced in Windows 2000. What I meant was that Windows Vista actually USED junction points (Documents and Settings, etc).

A. Skrobov
28 Sep 2009 10:51 AM
#

["Why does Explorer show me a file where all operations fail except delete?" -Raymond]

Explorer has no problem showing files like 'con'; on these all operations fail including delete.

someone else
28 Sep 2009 10:53 AM
#

“Explorer has no problem showing files like 'con'; on these all operations fail including delete.”

I believe this is one of those “two wrongs don't make one right” situations

John
28 Sep 2009 11:00 AM
#

Technically, I believe the reserved names (con, lpt, etc.) are an artifact of Win32, not NTFS. This discussion is about NTFS.

James Schend
28 Sep 2009 11:48 AM
#

Karellen: The obvious point you're missing is that if Windows added a UI for hard links, *non* techno-geeks will run across it and be confused, have broken backups, etc etc.

Hell, I consider myself a "techno-geek" and I've screwed this up before-- specifically, I created a backup script using robocopy in Vista that recursively copied the same symlinked folder inside of itself over and over again. (This can happen when copying your user directory in Vista if the robocopy script is running as Admin-- run the script as User and permissions prevent the problem.)

The target consumer of Windows and (most) Unixes is very, very different.

Besides, you have to do a cost/benefit analysis here. Raymond has laid out all the costs-- now what's the benefit to having access to hard links?

I can't think of a single thing I'd use them for. Can you?

Aneurin Price
28 Sep 2009 11:56 AM
#

[dup. -Raymond]

The linked thread, so far as I can see, doesn't actually provide an answer to the question.

However, some testing indicates that, although 'C:\Documents and Settings' is inaccessible, 'C:\Documents and Settings\<username>' is the same as 'C:\Users\<username>', so I imagine the answer lies somewhere in that direction. I guess it's a link that's been special-cased to be unreadable by everyone in order to discourage its use?

[*The answer is in the queue. I'm just calling out that it is a duplicate request. Repeating a question makes me less likely to answer it, but in this case, he lucked out because the answer was already on its way. -Raymond*]

someone else
28 Sep 2009 12:10 PM
#

Speaking of Robocopy, while it's nice that it has a switch to turn off deep junction copying, it would have been even better to add a switch to recreate the junction point, if possible (i.e. on the same volume). *sigh*

Random832
28 Sep 2009 12:15 PM
#

Let's keep in mind that Unix/Linux GUI file managers don't have an interface for creating hard links - at least they didn't the last time I used one.

WndScks
28 Sep 2009 12:18 PM
#

fsutil requires you to be admin for no reason, so you can't even create hardlinks without 3rd party tools on XP (Hey MS, how about testing things as non admin)

[*fsutil has a very good reason for requiring administrator privileges: It lets you change administrative settings for a drive. It does more than just create hard links. -Raymond*]

WndScks
28 Sep 2009 1:00 PM
#

@Raymond: And clearly the admin check should only be called when you are about to do something that requires admin rights (Or don't check at all and fail with a access denied error)

[*I'm sure the people responsible for fsutil have that on their list. It just needs to be prioritized against all the other things to be done. -Raymond*]

Gabest
28 Sep 2009 1:23 PM
#

NTFS file streams are rarely used too, but I do create files with streams in my programs and that explorer cannot see them or at least include their size in the total is annoying.

[*Imagine the confusion that would result if "Size" included the sizes of alternate streams. (Also consider how slow this would make directory enumeration.) -Raymond*]

someone else
28 Sep 2009 1:28 PM
#

@Gabest:

What do you do that for??

(Are you the Gabest of video software fame?)

jabelli
28 Sep 2009 1:58 PM
#

For those wondering what shell extension "someone else" was talking about, he probably meant this one: http://schinagl.priv.at/nt/hardlinkshellext/hardlinkshellext.html

someone else
28 Sep 2009 2:03 PM
#

Yes, I was. I trusted Google on this one.

Alexandre Grigoriev
28 Sep 2009 2:19 PM
#

@Gabest,

Explorer uses alternate streams to tag files with custom attributes. For example, you can tag a plain .txt file with them.

feddy
28 Sep 2009 2:59 PM
#

"If a file has multiple names, editing the file by using different names simultaneously may result in data corruption.

If a file has multiple names, all the names must exist on the same drive. "

These are logical consequences of what a hard link is and shouldn't need to be stated explicitly.

someone else
28 Sep 2009 3:04 PM
#

@feddy:

Experience tells us that a warning too many is better than a warning too few.

Nick
28 Sep 2009 3:28 PM
#

Well, hey, I'm famous!

I'd completely forgotten about asking this question until I saw my name there -- that's what over 3 years will do I guess :)

Honestly, I expected the answer to pretty much come down to (even experienced) user confusion, but at the same time wondered if there were more technical reasons involved in the design decision.

I can easily see not enabling an interface to hard links on consumer versions of Windows, but there are many potential uses of hard/symbolic links on Windows Server. I've only used them a few times, but it would be nice for shell integration to make that easier (though Vista/Win7/Server08 are somewhat better in that regard).

Anyway, that's for answering my question!

Blake
28 Sep 2009 6:57 PM
#

Certainly you can use fsutil.exe, linkd.exe and junction.exe, but with Vista or later it's easiest to use the cmd.exe builtin 'mklink' for hardlinks, and both common sorts of junction points.

For bonus points, don't forget that junction points are an extensible infrastructure and third-parties can create their own types of junction points. Explorer would be hard pressed do anything sensible there.

Tom Finnigan
28 Sep 2009 6:59 PM
#

So, does that mean that explorer has UI for creating symbolic links? I've been using mklink, but a GUI would be nice.

Barry Kelly
28 Sep 2009 7:20 PM
#

The reason backup programs fall over is because there is no UI for creating hard links, so the issues aren't found in testing; and a similar situation with confused users. It's a circular problem.

I for one use hard links all the time, with Cygwin ln and cp -l, but I'm not under any illusions about average users' familiarity with the concepts. tar and other utilities, and the backup program I use (rdiff-backup) are aware of hard links.

someone else
28 Sep 2009 7:21 PM
#

Well, look at that: http://technet.microsoft.com/en-us/library/cc766301(WS.10).aspx

“This user right should only be assigned to trusted users. Symbolic links (symlinks) can expose security vulnerabilities in applications that aren't designed to handle symbolic links.”

This might explain why there's no explorer interface. It's even more of a fringe case.

Dan
28 Sep 2009 7:41 PM
#

"There was a Windows XP PowerToy for creating hard links, but the backup team recommended that it not be released because so many third party backup programs exploded when they stumbled across files that had multiple names in the file system."

Let's not forget about filesystem scanning/copying programs and NTFS junctions. Plenty end up in infinite recursion when you have a junction that points to an ancestor. Either they run out of address space and die or hit a character limit for their paths and finally start unrecursing.

Dan
28 Sep 2009 7:47 PM
#

@Aneurin Price Obviously you get access denied when accessing it directly so that aforementioned backup programs can't recurse into it and cause the problems mentioned in the article.

But legacy programs with poorly coded paths can still use them to directly access files or folders, as you indicated.

Blake
28 Sep 2009 7:57 PM
#

Ugh - s/junction point/reparse point/g in my last comment. Junctions and symlinks are both sorts of reparse points.

Eric TF Bat
28 Sep 2009 7:57 PM
#

I think you've got the cart before the horse. Hard links are hard to understand because Explorer doesn't support them, not the other way round. The solution is obvious: Raymond, you must go down to the basement of the mysterious Building 13, borrow the official Microsoft Time Machine (R)[TM], go back to when Windows was first invented, and get them to put hard link support into whatever Explorer was called in those days.

See? Easy solution. I don't know why you complain so much.

Sam
28 Sep 2009 8:07 PM
#

"Why doesn't Explorer have a UI for hex-editing a file?"

That would be pretty handy! Did it make it into Windows 7?

(umm.. joking, in case that's not obvious.)

Yuhong Bao
28 Sep 2009 8:51 PM
#

Also for historical reasons. Windows 95 did not support hard links since it only supported the FAT file system, and even in the early days of NT it was primarily used for the POSIX subsystem which needed them.

hexatron
28 Sep 2009 10:05 PM
#

I didn't find out that NTFS supports hard links until about two years ago. Then I created a few to play with, and noticed the confusion they caused.

Then I remembered they weren't particularly useful in Unix either, but were integral to the directory structure in ancienter file systems (. and .. are hard links to the current and parent directories, etc)

So I forgot all about them again.

Falcon
29 Sep 2009 12:02 AM
#

"Hard links" in FAT were reported as cross-linked files by file system checking utils. This was the correct action to take, since they weren't supported by the file system, therefore their existence would have created problems - inconsistency between multiple directory entries, deleting one would free the cluster chain and destroy other "copies", etc.

I can understand the confusion even among advanced/experienced users. I guess they have a certain unnatural feel, as two separate files would be expected to be independent at the file system level. Even when you know about the concept of links (both hard and symbolic) but don't know that a particular file is a link, it's easy to fall into a trap.

Teo
29 Sep 2009 1:05 AM
#

This is very much off-topic, but still very interesting. Why are "format.exe", "chkdsk.exe", "fsutil.exe" and so on manifested "asInvoker"? They clearly require admin privileges, because they can kill every bit of information on your system.

And yes, I still remember the fun when my backup application "discovered" the Vista junctions inside user profile dir and merrily crashed :) Luckily, MSDN had the correct code for detecting them and I could fix the bug in under 10 minutes.

Worf
29 Sep 2009 1:40 AM
#

Hard links are fun!

On UNIX, they're a great source of security vulnerabilities. Thankfully, their limitation of being able to live on one filesystem meant their use is pretty restricted, but even in geek-land, I end up having to test behaviors. Soft links are much easier to use, and most people use them for they get rid of many hard link issues, though they are slower because of it.

About the only major use of hard links I can tell is in OS X's Time Machine feature, where identical unchanged directories are hard linked to save space.

Drak
29 Sep 2009 1:41 AM
#

Sorry if I'm asking a stupid question here, but what is the gain in a hard link over a soft link?

anonymuos
29 Sep 2009 2:05 AM
#

But why doesn't Explorer not traverse hard links, SMB symbolic links OR junction points when double clicked? (As I see, programs can traverse it but the file manager cannot navigate thru hard links?) Not under any obligation=unsatisfactory rude lazy arrogant crippled answer. Why Explorer can't calculate the correct folder sizes if it contains hard links (e.g. WinSxS)? Explorer is a big FAIL beginning with Vista. I was a total Windows zealot since Windows 95, but I was so disappointed with the shell in Vista that I plan to forever stick with XP until some OS with a full-featured non-crippled shell comes along. You have no right to take away major features people used. If some program or the other in the OS doesn't expose every NTFS feature (http://en.wikipedia.org/wiki/NTFS#Features), then who will? Why are you making the shell super-dumbed-down? Couldn't people work with the pre-Vista shell? I really expect to see Explorer at least navigate to the pointing folder in Windows 8 when a junction point is clicked instead of throwing up "Access denied". I am neither a developer nor an IT pro, just an end user but I find Explorer horrible non-customizable and dumbed down beginning with Vista. Just how many things Microsoft has never put because it would too nerdy?

[*What is the "correct" size of a folder containing hard links? (Psst, all folders contain hard links. Yet another person who doesn't understand hard links.) -Raymond*]

porter
29 Sep 2009 2:25 AM
#

>> Let's keep in mind that Unix/Linux GUI file managers don't have an interface for creating hard links - at least they didn't the last time I used one.

Let's keep in mind that that's a broad claim on little evidence.

Wayne
29 Sep 2009 2:44 AM
#

>> But why doesn't Explorer not traverse hard links, SMB symbolic links OR junction points when double clicked?

Does for me in XP. Explorer treats the junction point/symlink is a regular directory -- I'm not even sure it knows the difference.

I will agree that Vista's explorer is terribly hard to use when compared with XP's.

Karellen
29 Sep 2009 2:54 AM
#

@porter: As one datum, Dolphin from current KDE (4.3.1) does not have an interface for creating hard links. It can copy, move and create symbolic links to files, yes. Hard links, no.

Var
29 Sep 2009 3:06 AM
#

Nitpick: Features in a large market leader that dislikes competitive innovation start at minus 100 points.

Features in smaller disruptors just get implemented.

someone else
29 Sep 2009 5:52 AM
#

“Hard links are hard to understand because Explorer doesn't support them, not the other way round.”

Ah, you also clicked cancel after “When you create a hard link...”

“Why are "format.exe", "chkdsk.exe", "fsutil.exe" and so on manifested "asInvoker"?”

Because they aren't supposed to be used by the common folk anyway. Ask a non-computer-savvy person what these commands do.

Dee
29 Sep 2009 6:41 AM
#

To Teo:

"This is very much off-topic, but still very interesting. Why are "format.exe", "chkdsk.exe", "fsutil.exe" and so on manifested "asInvoker"? They clearly require admin privileges, because they can kill every bit of information on your system."

What if you want to format your USB key or 1.44" floppy?

someone else
29 Sep 2009 6:48 AM
#

What's a 1.44” floppy? But seriously, corporate admins usually don't like it when the users introduce their own portable media, so this is more of a problem for home users ... who run as admin or in easy reach of admin privileges.

Anonymous
29 Sep 2009 6:50 AM
#

It's for the same reason why the only text editor in default windows is notepad and the only image manipulation program is paint. Look at Mac or Linux and compare yourself.

Simply speaking, Windows sucks in any way.

Anonymous Coward
29 Sep 2009 7:40 AM
#

Good point, Anonymous, but I must for the sake of fairness point out that most of the good stuff that comes with GNU/Linux is available for Windows too.

TC
29 Sep 2009 7:41 AM
#

@hexatron: You say hard links are not particularly useful in Unix either?

They're actually useful in Unix *and* Windows. A backup tool can create a disk copy where, if a file hasn't changed, its entry is a hard link to the previous version. Thus, each disk copy looks and works like a full disk copy, but in reality, it only allocates space for the files that actually changed since the previous copy. This is the best of both worlds: utility and efficiency.

I tried to get this going on my Windows laptop, using rsync from a Ubuntu "Live CD" to make backups on a USB drive. (Ubuntu includes an NTFS driver.) But it wasn't really acceptable, because when I viewed the backups later from Windows, I couldn't see which files were "real", and which were hard links. This made it difficult to test the process, and be confident that it was working correctly.

Could NTFS maintain a reference count for each file? (ie. the number of directory entries that point to that file data?) Then, Explorer could use that count to highlight entries pointing to files with reference counts greater than 1. The highlight would effectvely mean: "This file has several different names. See 'hard linked files' for more information".

André
29 Sep 2009 7:51 AM
#

@jabelli

LSE is cool. I'm using this for years :)

@anonymuos (Tuesday, September 29, 2009 2:05 AM)

good questions. the size detection when HardLinks are used is still broken. People only see that the WinSxS folder is soooooo huge, but don't know that the files are linked to the target folder (C:\program fils;C:\Windows;C:\Windows\System32). This is disappointing, that it is not easy to see in Explorer if the file is a hardlink or not.

I've submitted this to Vista/WIn7 feedback on connect, but you'll never get an answer from MS :(

someone else
29 Sep 2009 7:58 AM
#

@Anonymous:

I'm sure the antitrust people would have a field day if MS included a decent editor a image manipulation program. Like when they included a decent web browser.

Besides, when does the average person use Notepad or Paint? What do you propose as a replacement? Remember, it can't have functionality that would confuse the average home user (e.g. regexp is right out).

Karellen
29 Sep 2009 8:08 AM
#

"I couldn't see which files were "real", and which were hard links."

Uh, real files are hard links. The two are equivalent. That's the point. It's just that when you "create a hard link", what you're really doing is creating a *second* hard link to a file, just with a different filename from the first (original) link.

"Could NTFS maintain a reference count for each file?"

I presume it does. Otherwise, how would it know when all the filenames that refer to it have been deleted? What you want is just a bit of UI that gives a link count, which probably wouldn't be too hard, but might be confusing. Perhaps an extra datum in the "file properties" dialog, which already contains a bits of information people will skip over that they don't understand, might do here.

"it is not easy to see in Explorer if the file is a hardlink or not."

Once a hard link has been created to a file, there is no difference between the original name of the file and the new name of the file. It is impossible to tell which is the original. If either is the original. You could have created two extra hard links to a file, and then deleted the original directory entry. It's impossible to tell. Heck, file moves on the same file system are probably implemented internally as simply creating a new hard link to a file with the new filename and then deleting the old hard link with the old name.

@Worf : How, on UNIX, are hard links a great source of security vulnerabilities, as you claim? I've never come across this assertion before.

Gabe
29 Sep 2009 8:10 AM
#

If you really want to screw with a backup program, combine sparse files with hard links! "This directory will require 1e+38 bytes to back up."

TC: You may unfortunately be one of the technogeeks who doesn't understand hard links. Symbolic links work just as well as hard links for storing duplicate files. And there is no way to tell the difference between a "real" file and a hard link -- every directory entry on the disk is a hard link.

If you want to see the reference count, you can call GetFileInformationByHandle on the file (see http://msdn.microsoft.com/en-us/library/aa363788(VS.85).aspx for the data it returns).

TC
29 Sep 2009 8:22 AM
#

@Gabe: Huh? A symbolic link is am .LNK file. It's content is completely different to the original file. There's no guarantee that any given application will be able to open an .LNK file. *All* applications can open a hard linked file (if they can open a file at all).

Also, there *is* a way to tell the difference between a "real" file (ie., in casual terms, one with a reference count = 1) and a hard link (in casual terms, one with a reerence ocunt > 1). fsutil returns a unique number pertaining to the file data itself.

TC
29 Sep 2009 8:26 AM
#

@Karellen:

"I couldn't see which files were "real", and which were hard links."

- Uh, real files are hard links. The two are equivalent. That's the point.

Dude, read my post again. I understand what a hard link is. I undestand that *every* directory entry is a hard link. But in the scenario that I described, it's important to distinguish: "a directory entry that points to file data with NO OTHER directory entries", from: "a directory entry that points to file data with ONE OR MORE OTHER directory entries".

Is this a technical blog or what?

[*And then you'll have end-user questions like "Why do some of my files have a little asterisk next to them?" Answer: "If an asterisk appears next to a file, then that means that the file is accessible by more than one name. (Insert description of hard links here)." User: "Sigh. Windows is so hard to use. I should have gotten a Mac." -Raymond*]

TJ
29 Sep 2009 8:29 AM
#

Gabe: "You may unfortunately be one of the technogeeks who doesn't understand hard links. Symbolic links work just as well as hard links for storing duplicate files."

Notwithstanding whether or not TC understands hard links, symlinks absolutely would *not* work just as well for storing duplicate files in the scenario he mentions.

Unless you think a backup system whereby deleting a file from a 2-week-old backup means the file also becomes inaccessible from every subsequent backup, that is...

someone else
29 Sep 2009 8:36 AM
#

“Huh? A symbolic link is am .LNK file.”

No. Do the freakin' research.

Paul M. Parks
29 Sep 2009 8:40 AM
#

These comments appear to be proving Raymond's point rather nicely.

Tim Gradwell
29 Sep 2009 8:42 AM
#

(in XP) If I drag the My Computer icon onto my start button I end up with an expanding My Computer icon at the top of my start menu. If I create a shortcut to My Computer and drag that onto my start button I end up with a shortcut to my computer. Is one of those a hard link and one a not-hard link?

someone else
29 Sep 2009 8:46 AM
#

@Tim Gradwell

My Computer is not a file, so it can't be hardlinked. Windows may be creating a folder with an appropriate desktop.ini for the expanding folder.

L.
29 Sep 2009 8:49 AM
#

@TC: A symbolic link is am .LNK file.

No. .LNK files are shell links, handled by the shell. Symbolic links (introduced in Vista) are reparse points, handled by the kernel.

Dean
29 Sep 2009 8:56 AM
#

"Nitpick: Features in a large market leader that dislikes competitive innovation start at minus 100 points.

Features in smaller disruptors just get implemented."

So Linux has every feature ever imagined already implemented? That's impressive!

sometwo
29 Sep 2009 8:57 AM
#

The whole rationale is so flawed. Then why don't you take out file-based shortcuts as well? Keep only the Windows 7 like taskbar behavior. Get rid of all *.lnk files for more dumbing down sake.

Daniel the Wise
29 Sep 2009 9:03 AM
#

Format can be run as regular user for Vista (for removable devices)

someone else
29 Sep 2009 9:04 AM
#

@sometwo:

Do the words “backwards compatibility” mean anything to you?

Aaron G
29 Sep 2009 10:16 AM
#

I've always looked at hard links as being a bit like tactical nuclear weapons. Under normal circumstances, you would never need them. Under exceptional circumstances where you think you need them, you probably still don't need them. If you really do need them, you probably don't know how to handle them properly. And if you use them anyway, there's a very good chance that you'll cause an enormous amount of collateral damage, even if you know what you're doing.

It's probably for the best that there's no pretty UI for it, in the fashion of the big red "launch missiles" button you see in Hollywood movies that apparently doesn't do any checks or require any other input and is just begging to be hit accidentally or at the worst possible time. I'd be very surprised if there are any people out there who don't work for the Windows team and who have been able to use hard links to solve a non-trivial problem while adequately planning for all the potential corner cases.

Clinton Pierce
29 Sep 2009 11:02 AM
#

@Drak:

Mostly speed.

[disclaimer: I haven't used hard links in Windows, but have used them extensively in Unix. And I'm simplifying a *quite* a bit.]

When you open a file in Unix (/home/clintp/myfile) the system traverses the file system to find each component of the path (home in /, clintp in /home, and myfile in /home/clintp). Now that we've found "myfile" what you've got is an inode number -- because directories are essentially lists of names and inode numbers. That inode number can be used to find a structure (surprisingly called an inode) that contains a pointer to the actual data for myfile. That inode number-to-inode structure-to-data is just two reads -- very fast.

Normally for files* there's a 1-to-1 correlation between a directory entry an inode. Each directory entry for a file refers to one inode structure.

When you create a hard link you create a second directory entry somewhere in that filesystem that refers to the *same* inode structure. Using either name you're 1 step away from getting to the data.

Think of a symbolic link as a "special little file" that has another name inside of it.** When you open a file that's really a symlink the system has to take the target name and traverse the directory tree again to find the target file before it can get to an inode to find the data. And you can have symlinks pointing to symlinks. Following a chain of symlinks to get to the actual data takes a lot of disk reads.

* And by "normal" I'm going to mean files that have 1 name and 1 set of data that goes with the name. It's no more or less correct than any other arrangement, but represents the vast majority of files in a filesystem.

** Implementation has varied wildly over the decades. Sometimes the symlink target "name" can be in the directory entry itself, in the inode for the directory entry, or in a "file" that the inode refers to. They've all been done.

Kevin Eshbach
29 Sep 2009 11:09 AM
#

Going by memory I thought junctions were first introduced in the new version of NTFS which was part of Windows NT 4.0. There was a new API call introduced that allowed them to be created but there was no built in tool to do it.

Rick C
29 Sep 2009 11:28 AM
#

One interesting use of hard links I saw on a Unix box was to protect database files. Create a directory, make hard links in it to all the DB files, and then set the perms on the links and the directory to 0000. Now you can't accidentally delete a DB extent.

Illuminator
29 Sep 2009 11:34 AM
#

Here's a more pressing question of the ages:

Why, 15 years later, does Windows Explorer still REFUSE to let me sort by file extension like every other normal file manager?

someone else
29 Sep 2009 11:40 AM
#

File extensions are an ugly kludge that should have been abolished a long time ago but refuses to die. You can sort by type, which makes somewhat more sense.

Absotively
29 Sep 2009 11:50 AM
#

@Illuminator:

It lets you sort by type. I'm not sitting at my Mac right now, so I can't confirm this, but I think that's the same thing OS X's file manager does.

I'd prefer sorting by extension, but this fits better with the default of hiding extensions. (Of course, I'd also prefer the opposite default there, but like everyone else here, I'm something of a techno-geek.)

Paul M. Parks
29 Sep 2009 11:52 AM
#

Illuminator: File extensions are hidden by default in Explorer, but there is a "Type" column that shows the type associated with the file. Having a separate column named "File Extension" would be confusing on a number of levels for the average user that has not enabled display of extensions.

David Walker
29 Sep 2009 11:53 AM
#

@anonymuos with the weird spelling: Annoyed much? "You have no right to take away major features people used."

If "You" means Microsoft, they can do (almost) anything they want.

And I like how the very first comment asks Raymond to go "find out" something. It's great to know that he is at all of our beck and call...

Karellen
29 Sep 2009 12:23 PM
#

@Rick C: Under Unix-style systems the permissions of a file are stored with the file, not the directory entry/hard link. If you create a second hard link to a file and set its permissions to 0000, then the permissions of the *file* are 0000 - i.e. no-one can read/write the file from either link (except root).

I suppose creating extra hard links to the files somewhere out of the way might prevent against accidental deletion by anyone who has the privs to do so (who has permissions to access/delete DB files other than the DB daemon UID?) - but why not just keep the only copies of the files in that out of the way location?

Doesn't make much sense to me. But I could be missing something...

rs
29 Sep 2009 12:46 PM
#

Alternate folder names, namely "." and "..", were confusing even in DOS. FindFirstFile/FindNextFile still report these, by the way.

John
29 Sep 2009 2:56 PM
#

@Illuminator: If it bothers you that much why don't you create a column handler? http://msdn.microsoft.com/en-us/library/bb776831(VS.85).aspx

Random User 43790
29 Sep 2009 3:11 PM
#

Given that most users appear to assume the filename is directly and inseparably attached to the file, it is unsurprising that more than a few are confused by "these 'hard link' things".

That hard links are/resemble (implementation specific) pointers causes problems, too.

I once saw a study (no link; look around) that said most (but not all) people can comprehend conceptual variables ["this" can contain "that"], but a depressingly large portion of that group have significant problems comprehending conceptual pointers ["this" leads you to "that"]. (Note: the topic was the conceptual [imaginary/computer/etc.], not the physical versions of these. People had fewer problems understanding boxes and phone books.)

Nick
29 Sep 2009 3:53 PM
#

@someone else:

The problem with file extensions is that nobody has proposed a better way to indicate a file's contents.

I like file extensions. I can see at a glance, without running another program, if a file is a picture, a text document, or executable. I can send a file to another operating system and that metadata is still available. Permissions, alternate data streams, file date/times and almost every other intrinsic property of that file may be gone, but the most important one, the file's type, is still there.

Contrary to *nix fanboy belief, file extensions are not bad.

Karellen
29 Sep 2009 6:31 PM
#

"I like file extensions. I can see at a glance, without running another program, if a file is a picture, a text document, or executable."

No, you can't. You can see what the person who named it *thinks* it contains, or what they want you to think it contains, but not what it actually contains.

To find out what it actually contains, you just need to use a little magic(5)

(<http://linux.die.net/man/5/magic>)

Wayne
29 Sep 2009 8:01 PM
#

Sometimes what someone thinks it contains is more important? What's the difference between file.txt, file.c, file.php, file.asp, file.sql, file.tpl?? They're all just ASCII files! The extension provides a level of meaning.

Miral
29 Sep 2009 10:37 PM
#

Not sure if this is getting a little off-topic or not; the other day I was trying to use mklink to set up junction points for the Program Files folders on Win7 (since I dislike those names and would much rather have them be called "Programs32" and "Programs64"). It's a lot more painful than it should be, especially since the original's permissions don't carry across to the junction. I've managed to copy most of the permissions over via cacl, but I can't work out how to change the owner of the junction to TrustedInstaller (to match the original). Neither console tools nor the GUI seem to want to admit that that account exists.

(All I wanted to do was to rename them, while keeping the original name valid for compatibility. I wasn't even trying to put them onto a different drive -- that would have been better, but I've read the horror stories that causes due to transactional NTFS.)

Rick C
29 Sep 2009 10:50 PM
#

Karellen, perhaps I misremembered the exact details and it was merely the directory that had 0000. Then nobody[1] would be able to get into the directory to delete the hard links, which would preserve the files.

"(who has permissions to access/delete DB files other than the DB daemon UID?)"

Unless you've experienced it, you'd be horrified how many places don't get that right.

[1]obviously anyone with root access can override this.

Nick
29 Sep 2009 11:08 PM
#

@Karellen

I'm very familiar with magic numbers and `file`, and I see them as an even worse solution to determining file type than file extensions. As pointed out, there are many, many different forms of ASCII text documents. There are times when `file` fails at correctly determining file types. Finally, you are now stuck using a system-dependent binary to decide what files are what?

Yes, file extensions can be changed arbitrarily, but in a way that's the point. They can be abstracted away if you want (as Windows does by default) or you can make use of them. The most important thing is that all files with a given extension/type are handled the same way.

I've told Windows I want PHP source files to open in Notepad++ and to use an icon I created. That's easy to do with file extensions. How would you suggest doing that with magic(5) and `file`?

(I realize this is completely off-topic. Sorry. But hey, a bunch of people having off-topic discussions is the sign of a popular and successful website. Congrats Raymond!)

Drak
30 Sep 2009 1:49 AM
#

@Clinton Pierce

Thanks for that answer :)

T
30 Sep 2009 2:09 AM
#

@L:

".LNK files are shell links, handled by the shell. Symbolic links (introduced in Vista) are reparse points, handled by the kernel."

Thanks, I didn't know that. But I had XP in mind, which explains the confusion.

Another poster pointed out that symbolic links would not be satisfactory in the scenario that I described. So I stand by my original comment, that hard links *are* useful in appropriate scenarios. It would just be nice if explorer could distinguish files with multiple links, for reasons I explained before.

TC
30 Sep 2009 2:13 AM
#

(edit: Last post @L was from TC. Not sure where the "C" went!)

Gabe
30 Sep 2009 3:10 AM
#

Aneurin Price: It's made unreadable with just a regular Deny ACE that prevents you from reading or deleting it. It has nothing to do with it being a symbolic link.

Presumably they don't want you reading it so that you don't write new apps that depend on it. And many users are likely to try to delete it if they can't read it, so maybe that's why they won't let you delete it either.

porter
30 Sep 2009 3:47 AM

#

>> The problem with file extensions is that nobody has proposed a better way to indicate a file's contents.

Type/Creator, MIME types, Content-Type, extended attributes and file headers.

I've heard people say that to turn a file into a zip file you just highlight the file and change the extension.

AC
30 Sep 2009 4:50 AM
#

@anonymuos

"But why doesn't Explorer not traverse hard links ... (As I see, programs can traverse it but the file manager cannot navigate thru hard links?)"

You're nearly there... Explorer has no problems traversing hardlinks. It can't /read/ the folders you're talking about (which just happen to be hard links - EG have more than exactly 1 reference) because the 'read' permission on them is set to deny.

And of course, it's down to appcompat:

Application Compatibility: Junction Points and Backup Applications

"... These junction points have file attributes of FILE_ATTRIBUTE_REPARSE_POINT and FILE_ATTRIBUTE_SYSTEM, and the access control lists (ACLs) must be set to ""Everyone Deny Read". Applications must have permissions in order to call out and traverse a specific path. However, enumerating the contents of these junction points is not possible."

http://msdn.microsoft.com/en-us/library/bb756982.aspx

And I imagine in Windows 8 or later they'll be removed completely. The problems you're experiencing are probably designed to get you used to the 'new world order', without immediately breaking things which reference %SYSTEMDRIVE%/Documents and Settings/%USERNAME%/.MyApp/ (for example).

It's not easy. That's why I tend to just read and not post because I'll probably get something wrong. I just happen to have dealt with these recently (cf. Applying an incorrect security template which then led to an infinite loop backup situation. With the perms that Microsoft has on them, that didn't happen) so I know a bit about them.

Looking forward to Raymond's post on them. Does that have a planned date? I assume that date is likely to change if you (Raymond) do a non-queued post anyway.

John
30 Sep 2009 6:02 AM
#

@porter:

Everything except file headers are stored independent of the file. What happens when file is copied to another file system? Uploaded to the web?

And file headers only work so well. A program to truly identify the file must know how to read the header (or the magic number, or whatever). Text files store a wide variety of different formats (as already mentioned) and they don't have headers.

Whether it's perfect or not, file extensions work amazingly well. And as to someone renaming a file extension to seemingly zip the file - well, there *is* a certain degree of knowledge/thinking users must bring to the table. Renaming a file from .TXT to .JPG won't make it an image, renaming a file from .TXT to .ZIP won't zip it either. This isn't a hard leap for most users to make.

Karellen
30 Sep 2009 7:42 AM
#

Yes, file(1) is terrible at telling the difference between different types of text file. Just watch:

```
karellen@host:~$ ls

file-1  file-2  file-3  file-4  file-5  file-6  file-7

karellen@host:~$ file *

file-1: ASCII text

file-2: HTML document text

file-3: RCS/CVS diff output text

file-4: Bourne-Again shell script text executable

file-5: ASCII C program text

file-6: ASCII make commands text, with escape sequences

file-7: a /usr/bin/perl -w script text executable

karellen@host:~$ file -i *

file-1: text/plain; charset=us-ascii

file-2: text/html; charset=us-ascii

file-3: text/x-diff; charset=utf-8

file-4: text/x-shellscript; charset=us-ascii

file-5: text/x-c; charset=us-ascii

file-6: text/x-makefile; charset=us-ascii

file-7: text/x-perl; charset=us-ascii

adam@adam-1:~/tmp$
```

How on earth could any application possibly tell the difference between these file types?

Stuck with a system dependent binary? Well, kind of, in that you can't just copy the file(1) binary from, say, FreeBSD to Windows64 and make it run. I mean, the source code is pretty portable and can be compiled on almost any platform, with Windows ports available <">http://gnuwin32.sourceforge.net/packages/file.htm> but I guess that doesn't count, huh?

As for getting Windows to use certain programs based on various file types, I guess I'd use the portable <">http://linux.die.net/man/3/libmagic> (the core of file(1) in a handy to reuse shared library) to determine the correct MIME type of a file based on its contents, and then allow users to specify their preference of application to use with varying MIME types.

Yeah, that might work.

[*Well, except that file-5 was actually a C++ file, not a C file. And file-1 was actually a LitWare Log File. -Raymond*]

porter
30 Sep 2009 9:14 AM
#

>> Everything except file headers are stored independent of the file. What happens when file is copied to another file system? Uploaded to the web?

I would argue the filename is not part of the file. The stream of bytes is the file, and the file may have zero or more names that reference the file (zero because a file could be open and then the last name unlinked). Renaming a file does not change the type of data.

As we have been saying the same stream of bytes can have two filenames refering to the same file, both of which could have different extensions or even no extension.

In NTFS the streams associated with the file are part of the file so writing the MIME type to the stream would stay with the file. When transferring the file with HTTP the file type lives in the header field "Content-Type".

someone else
30 Sep 2009 9:56 AM
#

By the way, OS/2 actually stored the file type independently of the actual extension. It never really caught on, though ...

Gabe
30 Sep 2009 10:42 AM
#

Raymond, don't be silly. If file-5 was really a C++ file, it would contain a "//" or the words "template", "virtual", "class", "public:", or "private:". If it were a Java program, it would contain the word "import".

What happens if your programming language isn't popular enough to be among those half-dozen that libmagic knows about (like JavaScript, C#, and VB)? Just figure out a way to tell the difference between your language and one that it already knows about, add it to names.h, compile, submit your patch to the maintainers of the file, and wait for the rest of the world to update their library! What could be simpler?

someone else
30 Sep 2009 11:16 AM
#

(Almost) every valid C file is also a valid C++ file.

Yuhong Bao
30 Sep 2009 11:26 AM
#

[And then you'll have end-user questions like "Why do some of my files have a little asterisk next to them?" Answer: "If an asterisk appears next to a file, then that means that the file is accessible by more than one name. (Insert description of hard links here)." User: "Sigh. Windows is so hard to use. I should have gotten a Mac." -Raymond]

Interesting, since Mac OS X, being based on Unix, has hard links as well. But, yes, indeed I know of no file manager that does this, even on Mac or Unix.

porter
30 Sep 2009 12:13 PM
#

>> But, yes, indeed I know of no file manager that does this, even on Mac or Unix.

Have you never heard of "ls"? If you use the long listing "ls -l", the 2nd column, right after the attributes indicates the number of links to the file.

```
$ ls -l wibble

-rw-r--r--   1 porter  porter  1 Oct  1 05:11 wibble

$ ln wibble wibble2

$ ls -l wibble*

-rw-r--r--   2 porter  porter  1 Oct  1 05:11 wibble

-rw-r--r--   2 porter  porter  1 Oct  1 05:11 wibble2
```

Paul M. Parks
30 Sep 2009 12:23 PM

#

Gabe: I write a phenomenal amount of C++ modules that contain none of the symbols and keywords you listed. And yes, they are C++ modules, not C. You'd have to compile the code to tell the difference. I think I prefer an extension.

Paul M. Parks
30 Sep 2009 12:47 PM
#

Gabe: Or perhaps you're being ironic. At least, I hope so.

Nick
30 Sep 2009 3:25 PM
#

@Karellen

I don't disagree that `file` does a pretty good job at describing a lot of different types of files. The two problems that jump out immediately are:

- What I said earlier (and Raymond gave great examples for).

- `file` is constantly playing catch-up with all the various kinds of file formats out there. Some might say that this means you've already lost the battle.

- File associates are much cleaner with extensions:

In Windows I can double-click a *.tda3 file and have the TMPGEnc project open in the correct program. What are the odds that `file` knows about that (or other less well-known types of files)?

I have many *.xpi, *.wsz, and *.zip files. Each of these is just zip file, but in Windows they will open with the correct program (Firefox, Winamp, and WinZip for example). There's no way for a `file`-empowered shell to do this.

anonymuos
30 Sep 2009 5:05 PM
#

@David Walker, of course they have the right. But remember we also have the right to buy or not buy. It implied that. Most understood it.

@Illuminator and @John,

there's already a column handler for extensions at http://www.xrayz.co.uk/extension-column/. But the horrible Vista/Windows 7 Explorer removed it without providing a proper replacement (Windows Search and Property system? No sorry)!!!!!!

Stefan Kanthak
30 Sep 2009 7:58 PM
#

@Raymond:

[And then you'll have end-user questions like "Why do some of my files have a little asterisk next to them?" Answer: "If an asterisk appears next to a file, then that means that the file is accessible by more than one name. (Insert description of hard links here)." User: "Sigh. Windows is so hard to use. I should have gotten a Mac." -Raymond]

Have you used shell links in Windows 95 and later? They show an arrow as overlay icon.

Have you ever used the CD burning extension provided with XP and later?

When you "write" one or more file(s) to CD they show up with a overlay icon resembling a "dagger" as long as they are sitting in the staging area. And IIRC the shell uses hardlinks in this case.

NB: if you "write" a directory to CD then the shell copies that directory with its contents to the staging area... and no overlay icon shows!

[*The difference is that you can explain "it's a shortcut" and "those are files waiting to be burned to CD" in 10 words or less, and people will say "Oh, okay" and not "What kind of computer geeky technocrap are you talking about, and why should I care?" -Raymond*]

foxyshadis
30 Sep 2009 10:50 PM
#

Want to see if files have more than one hard link? Write or find a filesystem filter that modifies some property returned to every caller, or a shell extension that highlights or adds a column or property page or whatever you want. It's just that simple.

Teo
1 Oct 2009 1:05 AM
#

Actually, I found *once* a good reason to hardlink files. They were so severely fragmented, that NTFS just gave up every time someone tried to do anything besides reading from them (ERROR_FILE_SYSTEM_LIMITATION). These files were totally 40+ GiB. So I "copied" them for the sys admin to play with, while the programmers access them through the original names.

It was an interesting problem to deal with - I couldn't do anything with these files until I defragment them, *including* defragmenting them :) Which is described in this KB article - http://support.microsoft.com/kb/957180 but still fun.

Alexandre Grigoriev

1 Oct 2009 12:58 PM
#

@Gabe,

Having // is NOT a sign of C++ file. It's not your K&R C anymore.

porter
1 Oct 2009 1:27 PM
#

>> Having // is NOT a sign of C++ file. It's not your K&R C anymore.

I know many true ANSI C compilers that balk at //, and quite rightly so!

Alexandre Grigoriev
1 Oct 2009 2:15 PM
#

@porter,

True C99 ANSI/ISO compilers won't balk at //. It's not your C89 anymore.

Gabe
1 Oct 2009 3:02 PM
#

Paul, Alexandre, and porter: There is no reason that your C file couldn't have a // comment. My Borland C files had them in the early '90s. Even if you use strict K&R C, you could have a string or comment that has a // in it. In fact, libmagic will probably identify your JavaScript, C#, or Java package as C++ as well due to the // comments.

This was really an example of why using a ridiculous heuristic to guess a file's type is a much worse idea than letting the user assign the file type with metadata (an extension).

Paul M. Parks
1 Oct 2009 3:09 PM
#

Gabe: So it was irony; I apologize for jumping the gun. Your point is entirely correct. The idea of waiting for Explorer to scan the contents of hundreds of files in a directory to determine their types is a non-starter, anyway, even if the results were reliable.

Paul Reinheimer
1 Oct 2009 6:04 PM

#

The analogy I generally fall back on when it comes to hard links is lockers in a bus stop or high school.

When you create a file, you put a label on the locker, the contents of the file reside inside. When you create a hard link you place a second label on the very same locker. When you "delete a file" you pull a label off. Lockers with nothing in it are cleaned out.

The analogy holds through your various examples of temporary files (new locker, move label) and such.

It may be tricky to explain things, but it's far from impossible.

640k
2 Oct 2009 2:30 AM
#

With the inode indirection it's actually a locker in a locker. FAT is more direct, an directory entry points directly to the sector holding the content.

Stefan Kanthak
4 Oct 2009 11:30 AM
#

@Raymond:

[The difference is that you can explain "it's a shortcut" and "those are files waiting to be burned to CD" in 10 words or less, and people will say "Oh, okay" and not "What kind of computer geeky technocrap are you talking about, and why should I care?" -Raymond]

Let's try: a shortcut is an attributed pointer to a filesystem/shell object. When you open it, the shell resolves the indirection and opens the filesystem/shell object; but if that has been moved, then (depending upon certain registry settings) the shell will search it.

When the filesystem object is an executable, the shell link provides some parameters (options, working directory) for its call.

Is that explanation but complete?

Have I covered all the semantics attached to shell links?

NB: if you turn on SAFER with it's default settings don't forget to exempt *.LNK from the list of executable files.

Let's try to describe hard links:

A hard link is a(nother) name for an existing file. It can be placed anywhere on the same volume where the file resides. The several hard links which name the same file are all equal, there is no primary hard link. If you erase the last hardlink of a file the file will be deleted.

I omitted the NTFS specialties of hard links: they cache a part of the attributes of the file. When you change a file through one hard link and then run a "dir another_hard_link" you'll see the attributes BEFORE the change. But if you open that another_hard_link the attributes are updated. If you use the Windows Explorer you'll notice this when you enter a directory with hard links to files which Explorer reads to fetch the icon or tooltips or ...

[*I think the average user lost you at "attributed pointer". -Raymond*]

Cheong
4 Oct 2009 11:15 PM
#

To explain to developer is a bit easier...

First, imagine that you "new-ed" an object and stored it's pointer to a variable. Then you created another variable and copied the pointer to it.

Now, just think the "object" as the "file", "variable" as the "directory entry on filesystem" and the second vairable as "hardlink".

While this analogy may not cover all aspects of hardlink, this serves the purpose of explaining "what a hardlink is" well to any students who attended C++ programming class and doing well enough...

Yury Skaletskiy
5 Oct 2009 5:48 AM
#

This is obvious it's not necessary to expose such things in explorer.

I just wandering why it doesn't have a shortcut for Create Folder functionality. If you work from keyboard, you have to Alt-f, W, F when you create a new folder

[*? "Explorer doesn't have a shortcut for X. The shortcut for X is Y." -Raymond*]