The Old New Thing

How does Explorer detect whether your program supports long file names?

20 Oct 2004 7:00 AM | 70

When you register your program with a file association, the shell needs to decide whether your program supports long file names so it can decide whether to pass you the long name (which may contains spaces! so make sure you put quotation marks around the "%1" in your registration) or the short name.

The rule is simple: The shell looks at your program's EXE header to see what kind of program it is.

- If it is a 16-bit program, then the shell assumes that it supports long file names if it is marked as Windows 95-compatible. Otherwise, the shell assumes that it does not support long file anmes.
- If it is a 32-bit program (or 64-bit program for 64-bit systems), then the shell assumes that it supports long file names.
- If it **can't find your program**, then the shell plays it safe and assumes that the program doesn't support long file names.

Note that third case. If you mess up your program registration, then the shell will be unable to determine whether your program supports long file names and assumes not. Then when your program displays the file name in, say, the title bar, you end up displaying some icky short file name alias instead of the proper long file name that the user expects to see.

The most common way people mess up their program registration is by forgetting to quote spaces in the path to the program itself! For example, an erroneous registration might go something like this:

```
HKEY_CLASSES_ROOT
   litfile
      shell
         open
            command
               (default) = C:\Program Files\LitWare Deluxe\litware.exe "%1"
```

Observe that the spaces in the path "C:\Program Files\Litware Deluxe\litware.exe" are not quoted in the program registration. Consequently, the shell mistakenly believes that the program name is "C:\Program", which it cannot find. The shell therefore plays it safe and assumes no LFN support.

Compatibility note: As part of other security work, the code in the shell that parses these command lines was augmented to chase down the "intended" path of the program. This presented the opportunity to fix that third case, so that the shell could find the program after all and see that it supported long file names, thereby saving the user the ignominy of seeing their wonderful file name turn into a mush of tildes.

And after we made the change, we had to take it out.

Because there were programs that not only registered themselves incorrectly, but were **relying on the shell not being smart enough** to find their real location, resulting in the program receiving the short name on the command line. Turns out these programs **wanted**

the short name, and doing this fake-out was their way of accomplishing it.

(And to those of you who are already shouting, "Go ahead and break them," that's all fine and good as long as the thing that's incompatible isn't something you use. But if it's your program, or a program your company relies on, I expect you're going to change your tune.)

Blog - Comment List MSDN TechNet

Comments

Jerry Pisk
20 Oct 2004 7:11 AM
#
If a program that my company relies on breaks because it does tricks like this you bet I'm going to change my tune - I'll start looking for an alternative. And since Microsoft isn't really willing to make sure programs like this don't survive I have no choice but to look at other platforms, that weed out crappy code.

Carlos
20 Oct 2004 7:18 AM
#
> The most common way people mess up their program registration is by forgetting to quote spaces in the path to the program itself!

The COM self-registration code in VB6 used to have this bug and it lead to many bizarre errors. For example, you'd see "D:\Program Files\Me\MyVBServer.exe is not a valid Win32 executable" in the event log. This would happen if someone had created a file called "D:\Program" which wasn't a valid win32 executable.

The security implications of this are also rather worrying.

Tim Smith
20 Oct 2004 7:23 AM
#
In other words, "Damn you Microsoft for making an operating system that doesn't break applications that were working just fine."

If I "broke" bad user data with each release of my software just because the user was "too stupid" to know that what he did was wrong, I would be out of business.

There is NO reason to assume that someone screwed up their registry entry just to make sure he received short file names. I would bet half of these people always wondered why they were getting short filenames but just thought it was "how it works".

Hey Raymond, in your experience have you run across a term used to describe situations like these? I have used the term "accidental contracts" to describe cases where a user, unbeknownst to himself relies of a undocumented behavior.

Gordon

20 Oct 2004 7:33 AM
#
It seems like users want this undocumented behavior pretty badly. Why not just have a registry key for it, rather than making ANY assumption? (Initially, I mean -- now you're pretty screwed.)

Thomas Heller
20 Oct 2004 7:37 AM
#
How do I mark a 16-bit Program Win95 compatible?
(We are still distributiong 16-bit exe. Once I had long filenames working, but someday it broke)

Raymond Chen
20 Oct 2004 7:42 AM
#
"Why not just have a registry key for it, rather than making ANY assumption? (Initially, I mean..."

I guess I don't understand. A registry key that says "Assume incorrectly registered programs are SFN?" That was the default behavior. Why do you need a registry key to set a behavior that is already the default?

And a registry key doesn't work once you have two programs, one that wants the setting on and one that wants it off.

Ben Cooke
20 Oct 2004 7:52 AM
#
Quite why a 32-bit application would want a short filename is beyond me right now. I guess it could just want a filename without spaces to avoid dealing with the quotes, but that's the only reason that comes to mind.

I dread to think what happens if the user installs such an application to a path like N:\apps\BrokenApp (where N: is perhaps some mapped network share) - now Windows will be able to find the executable and determine that it's 32-bit ... CRUNCH.

DrPizza
20 Oct 2004 7:56 AM
#
Why would I change my tune?

MS could write an appcompat shim (that could be used with any and all of these broken applications) that I could use to fix the problem.

It means that the core OS can work sensibly, broken applications can continue to run, and that their breakage becomes apparent, thus serving as a motivation to their vendors to fix them.

Raymond Chen
20 Oct 2004 7:58 AM
#

The problem is that the shim would have to be applied to *any program that uses shell32.dll* since it is shell32.dll that parses the registry key. Shimming litware.exe doesn't get you anywhere since it is not litware.exe that needs to alter its registry parsing code. It's shell32.

Miles Archer
20 Oct 2004 8:47 AM
#

Most people just want their computer to work and dont care how or why. Causing them avoidable pain is a great way to get them to switch to a platform that "just works".

Ben Hutchings
20 Oct 2004 9:17 AM
#

Carlos: There is some code in Windows XP that will warn you if you create a directory whose name matches up to the first space of the Program Files folder (e.g. "C:\Program" on a normal English version), though I don't remember when it runs. Creating that folder through Explorer doesn't trigger it immediately.

Brian
20 Oct 2004 9:42 AM
#

Will there ever be a day where Raymond describes some heroic workaround that MS undertook to keep some misguided app running that doesn't involve a debate on the merits of keeping said app running vs giving it some tough love? It's sooooo tedious.

Cooney
20 Oct 2004 9:52 AM
#

How about a registry key for the application that declares whether LFN support is desired?

Ari Berger
20 Oct 2004 9:54 AM
#

It seems unlikely to me that most culprits weren't "relying on the shell not being smart enough to find their real location". More likely they registered their programs incorrectly by accident. It still ran, so they didn't know they were doing somethng wrong. When they examined the pathnames that came in, they were short file names, so they wrote the programs to match.

Raymond Chen
20 Oct 2004 10:00 AM
#
"so they wrote the programs to match"

and therefore now rely on this behavior.

Gordon
20 Oct 2004 11:06 AM
#
Cooney got my meaning better than I expressed it -- the behavior I meant was "32/64-bit app that doesn't support LFN", rather than the tripartite assumption.

Andrew Taylor
20 Oct 2004 11:29 AM
#
Why can Windows find the file to execute it but not to check binary? Shouldn't that have been consistant from the start? If Windows gave enough feedback to developers that their registry entry was incorrect by not executing the file, perhaps even having the shell pop up a message box saying "Can't find c:\Program", this issue wouldn't have arisen in the first place.

Of course, crying about it now isn't productive, and I guess Windows takes the safest alternative given the choices.

Raymond Chen
20 Oct 2004 11:46 AM
#
The decision on how to parse unquoted strings in CreateProcess was made in Windows NT 3.1 (for compatibility reasons certainly). Imagine if NT 3.1 said "Sorry, lots of your 16-bit programs won't work. If you don't like it, go buy some other operating system."

Camillo
20 Oct 2004 1:49 PM
#
I really don't understand why people react this way. Raymond is spending his free time giving us a glimpse on how they were able to create the most succesful piece of software on earth... and you all begin to aruge that you would have done it better. Gentlemen, please show the evidences, case studies.
I prefer to stay silent and thank Raymond for sharing his know-how with us.

Jerry Pisk
20 Oct 2004 3:36 PM
#
Camillo, take a look at Linux. It barely existed 10 years ago, when Windows was first released. And look at where it is today, compared to Windows. Backwards compatibility

might be nice but is usually the worse choice when it's made at the expense of stability or security. And supporting lazy programmers only leads to more lazy programmers that won't bother to do things the right way.

I understand why Microsoft is doing all these things Raymond describes, but that doesn't mean I have to agree to it and that I can't say what I think about it, just because Raymond spends his time writing about it. Afterall, most of us do use Windows, why woudln't we want Microsoft to know what we think? Not all Windows users are VB draggers who do not understand how to quote paths…

Prasenjeet Dutta
20 Oct 2004 4:06 PM
#
Is it just me, or does anyone else have this LFN-not-supplied problem with regedit.exe on Windows 2000 (SP4 here)? I see a lot of "Are you sure you want to add C:\DOCUME~1\PRASEN~1\MYDOCU~1\foo.reg to the registry? y/n" because my Windows 2000 registry uses 'regedit.exe "%1"' for HKCR\regfile\shell\open\command. Wondering if _this_ was deliberate.

John Elliott
20 Oct 2004 4:15 PM
#
I tend to install my programs in C:\OPT rather than C:\Program Files. As Ben Cooke remarks, that's going to play havoc with a program relying on the above behaviour.

As for why I'd do such a thing: PROGMAN.EXE seems to have something of a blind spot about long filenames with spaces in them. You can set up a program item with such a name (either by browsing, or by drag/drop) but it won't run them. You have to convert the name to a SFN by hand, and it's then that you get the fun of working out whether "Program Files\Microsoft Developer Studio" corresponds to PROGRA~1\MICROS~1, PROGRA~1\MICROS~2 or PROGRA~1\MICROS~3. It's much simpler to install into C:\OPT\MSDEV where PROGMAN can see it straight away.

Rick C
20 Oct 2004 4:52 PM
#
"You have to convert the name to a SFN by hand, and it's then that you get the fun of working out whether "Program Files\Microsoft Developer Studio" corresponds to PROGRA~1\MICROS~1, PROGRA~1\MICROS~2 or PROGRA~1\MICROS~3."

dir/x is your friend when it comes to that.

Norman Diamond
20 Oct 2004 6:26 PM
#
1.
> If it can't find your program, then the
> shell plays it safe and assumes that the
> program doesn't support long file names.

Is that the reason why PhotoDraw gets registered without those quotation marks? The result is that when double-clicking a previously saved PhotoDraw file, Windows either can't find the PhotoDraw application or can't find the file you just double-clicked (I forgot which, it's been a while).

Surely the company that made PhotoDraw would have discovered this bug if they had ever tested the product -- after all, how can you miss what happens when you double-click a file in Explorer. No fix was ever released either. Will a shim be made to compensate for it?

2.
Does Windows 98 command.com support long file names? I copied and saved the following a couple of years ago because it was so striking. In between the two "move" commands, Windows made a random adjustment to its schizophrenia over the use of long filenames and short filenames. I didn't do anything else between those two "move" commands.

C:\My Documents>move an061.pdf ni
C:\MY DOCUMENTS\AN061.pdf => C:\MY DOCUMENTS\ni\AN061.pdf [OK]

C:\My Documents>move an017.pdf ni
C:\MYDOCU~1\AN017.pdf => C:\MYDOCU~1\ni\AN017.pdf [OK]

C:\My Documents>

3.
Does Windows Explorer support long filenames? Fixed in Windows XP SP2, not yet fixed in Windows Server 2003, on external hard drives. (Also can be fixed by grading to Windows 98, NT4, 2000, etc.)

Jerry Pisk
20 Oct 2004 6:47 PM
#
Another thing Raymond didn't mention - what's going to happen when there are no short file names? You can turn it off, for NTFS volumes, so if one does turn it off (as I do on my servers) programs not working properly with LFN break - and the vendor blames it on me, because their Windows run their ***** code just fine.

Simon Cooke [exMSFT]
20 Oct 2004 8:52 PM
#
John Elliot-

People still use Progman?

Bleuch.

Raymond Chen
20 Oct 2004 11:02 PM
#

"Does Windows 98 command.com support long file names?"

It does, but "move" is not part of command.com. It's an old DOS program.

"Does Windows Explorer support long filenames?"

I'm sure this is some sort of trick question.

Frank
20 Oct 2004 11:23 PM
#
I made the error described by Raymond, and now I can fix it. Thank you Raymond.

Chris Becke
21 Oct 2004 12:58 AM
#
I do find this concession to compatability a bit dodge. Installing these apps in a path without spaces causes them to break.

I remeber at one time trying to find out what % escapes the shell supported. I have a legacy app that needs to get the filenames passed to it winthout their extensions. I seem to recall finding things like %L would always pass the long name, but I never found any documentation describint the alternatives to %1 and %*

John Elliott
21 Oct 2004 3:14 AM
#
Simon Cooke: Ah, comrades, you may jeer and sneer, and you may scoff, but I've got into the habit of using it for all the apps that don't fit on my quicklaunch bar. Don't worry - I don't have it as my shell :-)

Chris H
21 Oct 2004 5:22 AM
#
Jerry:

> take a look at Linux. It barely existed 10
> years ago, when Windows was first released.
> And look at where it is today, compared to
> Windows.

Today, it barely exists, compared to Windows. And part of the reason for that is that people don't like dealing with the horrendous compatibility issues between applications and versions of core libraries.

DrPizza
21 Oct 2004 5:49 AM

#

"The problem is that the shim would have to be applied to *any program that uses shell32.dll* since it is shell32.dll that parses the registry key. Shimming litware.exe doesn't get you anywhere since it is not litware.exe that needs to alter its registry parsing code. It's shell32. "
Surely it'd have to be applied to litware.exe to mung its command-line arguments vector so that it got what it expected, rather than what it should have actually received?

DrPizza
21 Oct 2004 5:51 AM

#

"The decision on how to parse unquoted strings in CreateProcess was made in Windows NT 3.1 (for compatibility reasons certainly). Imagine if NT 3.1 said "Sorry, lots of your 16-bit programs won't work. If you don't like it, go buy some other operating system." "
This was in practice not that far from the truth anyway, so I'm not sure it would have been too much of a disaster.

mschaef
21 Oct 2004 6:50 AM

#

"If you remember Norton Navigator from 1995, it had a feature that let 16-bit apps "see" LFNs (not sure how exactly, I didn't use NN much). I'm curious if apps like NN cause the shell team any extra grief in app compat. "

I have no idea how this was done, but I have a guess I came up with at the time.

My theory is that Norton intercepted calls to the Win16 standard dialog APIs and displayed the Win32 standard file dialogs instead. I'm supposing that thunking and some creative systems programming was able to resolve the conflict bewteen Win16 and Win32. From that point, all that'd have to be done (I guess) is to ensure that the filenames passed back from the altered standard file dialogs match what a Win16 app would expect: short file names.

Of course, if apps expected to be able to do funky things to the standard dialogs (subclassing, new dialog templates, adding control, whatever), Norton would either have to disable its extension for that particular app or find a way to support the client app's expectations.

Does this sound plausible at all? It seems a lot more doable than somehow altering apps in place to handle long filenames correctly.

Matt
21 Oct 2004 7:37 AM

#

"take a look at Linux..."
Yes - a mediocre rip off of a 30 year old OS that's barely caught up to where the original was 10 years ago!
An OS that doesn't worry about spaces in program paths because most of its directories in the root file system have 3 character names. (Obviously 3 character file type extensions are arcane and detrimental to the user experience. The user would much rather interpret 3 character names for the most important system directories. I've

personally always rathered "/bin", "/usr/local/bin" and "/opt" to "Program files" as a place to put my programs!) Of course I'm being harsh - some of the more advanced directories have easily understandable 5 or 6 letter names.
And of course binary compatibility with old programs (through hacks in the OS) is obviously less important than a stable API and architecture. Users love to recompile programs and come to terms with the architecture of X. Windows 3.11 on DOS was obviously a dirty hack, but X on sh/bash/csh is such a clean answer to the GUI.
Users also love to learn dozens of different syntaxes for text configuration files. It's obviously superior to a consistent GUI. We'd much rather spend hours learning config file syntaxes than have our computers waste seconds loading and using a GUI.

David Candy
21 Oct 2004 8:30 AM
#
I presume %L forces a long file name rather than above rules. Can you force a short name (differently from above duscussion).

mschaef
21 Oct 2004 10:33 AM
#
" I've personally always rathered "/bin", "/usr/local/bin" and "/opt" to "Program files" as a place to put my programs!)"

Personally, I'd rather have /bin than "Program Files". /bin is _much_ easier to type, and for "Aunt Tillie", she's not going to care where the programs are installed anyway, given that Windows depends so heavily on installers distributing files across the disk (rather than just copying over program files manually).

Matt
21 Oct 2004 11:17 AM
#
If you're typing those places are almost certainly included in the path and there's path completion in both Linux and Windows at present.
If you're hunting around for something "Program Files" is a much more obvious place to look "/bin", "/opt" and "/usr/local/bin".
I'm not claiming Windows is perfect - theres no excuse for msimn.exe as a filename today, but it's still ahead of Linux.
And I'm responding to the argument that Linux has ditched its "unnecessary" compatibility weight more than Windows has for the sake of user and technological advancements.
Obscure *important* directory names (and configuration files and outdated GUI architecture) says it hasn't. In fact the only place Linux ditches compatibility is for binary executables. That's no great technical achievement and its a user nightmare.
If Linux were the revolutionary OS its proponents make it out to be those issues would have been addressed.
I'd be worried about giving such astounding insights away to Linux developers on an MS blog if I weren't sure they'd ignore them anyway. In 5 years time Linux developers will still be bitching about MS being anticompetitive. If Linux is technically superior it would have the market by now (ten years after widespread distribution). Their arguments will be even less compelling in 2009.

Josh
21 Oct 2004 12:03 PM
#
Program Files may be more obvious than bin, but why oh why couldn't microsoft have used Programs instead? That would have avoided much of the short versus long filename issues. Many of Microsoft's own commands still have issues with spaces being in a path or filename. Oh well!

Matt
21 Oct 2004 12:17 PM
#
A near good question, Josh, but I'm in an argumentative mood.
I think the reason (if MS considered that option) is that your "Programs" directory suggestion would conflict with the "Programs" start menu item. It's much better to have some distinction for the less technical. If "Programs" are what you click on the start menu then all those supporting files should be called something else. (I even include the exe as a supporting file given its shortcut often has a different name.) "Program Files" seems somewhat obvious after the Start Menu terminology is factored in.
Spaces and long filenames shouldn't have been an issue (outside of DOS/Win16/Win32[k?/s?/whatever?]) in 1995 let alone in 2004. MS has gone a fair way to making that so. Linux has not despite its proponents claims of technical superiority and superior adapatability.

mschaef
21 Oct 2004 12:34 PM
#
" It's much better to have some distinction for the less technical. "

Why are the less technical looking around in "Program Files" anyway?

The 95% case for modern users is: install, uninstall, repair an existing install, and change the components that are installed. The installer can do all of that from the "Add/Remove Programs" Control Panel. The cases I can think of that would require more access all imply enough sophistication that the difference between "Program Files", "Programs", and "bin" seems pretty trivial.

Just to put it in perspective, I don't remember having any trouble with odd directory names back when I was 12 and playing around with DOS on an 8088. If that's me with no formal training, very little informal training and not much experience with the machine at all, I have a hard time figuring out who exactly this user is who cares what's in "Program Files" and yet can't figure out a shorter directory name.

"Spaces and long filenames shouldn't have been an issue (outside of DOS/Win16/Win32[k?/s?/whatever?]) in 1995 let alone in 2004. MS has gone a fair way to making that so. Linux has not despite its proponents claims of technical superiority and superior adapatability. "

What specifically are you claiming that Linux isn't doing with spaces and long fime names?

mschaef
21 Oct 2004 12:53 PM
#

"If you're typing those places are almost certainly included in the path and there's path completion in both Linux and Windows at present."

Which I now have to obsessively rely on thanks to the choice of directory name.

"If you're hunting around for something "Program Files" is a much more obvious place to look "/bin", "/opt" and "/usr/local/bin". "

Sure, right. It makes _so_ much sense to look for include files under "/Program Files/Microsoft Visual Studio 2003/vc7/PlatformSDK/include" than "/usr/include".

Not to mention the fact that Windows installers have historically droped files all over the hard disk and not just in Program Files.

"Obscure *important* directory names"

Directory names that take a whole 10 minutes to figure out.

"(and configuration files"

Which are easily scriptable, easily transportable, and easily editable with simple tooling.

"and outdated GUI architecture)"

Which has done better remote access than Windows for coming up on 20 years now. Complaining about X Windows when the dominant function of most all windows systems these days is as a bitmap transport seems pretty pointless.

"In fact the only place Linux ditches compatibility is for binary executables. "

That's the side effect of a culture that has historically values openness, availabilty of source code and other information, and personal responsibility rather than corporate dominance.

With Linux, I know I can theoretically understand and fix what's going on in my computer. With closed source, it's just the opposite: I _have_ to rely on Microsoft/etc., who historically (and correctly) operates in its shareholder's interests. If I ever end up in a place where my interests diverge from Microsoft's and I depend on their software, then oh well... too bad for me: there's nothing _I_ can do, short of switching and hoping it doesn't happen again.

"If Linux is technically superior it would have the market by now (ten years after widespread distribution). "

The history of this industry is littered with the shells of companies that had technically superior products that got steamrollered by the competition.

It's not technical superiority that makes the decision, it's: can I get done what I need to get done. Since that's very much tied to things like file formats and software compatibility, the obstacles to Linux adoption shouldn't be underestimated by folks on either side of the fence. It will take a long time, if it ever happens, and to say that since it hasn't happened yet, it won't happen at all is to fundamentally misunderstand the motivations behind such a possible shift in consumer demand.

mschaef
21 Oct 2004 1:02 PM
#
FWIW, I do use Windows virtually _all_ the time, due to interoperability concerns. It just seriously concerns me that so much of my work is tied to one company, so dominant in its field, and my alternative choices are so limited.

余啊雷
21 Oct 2004 6:55 PM
#
Subversion » Obfuscation and the Long Filename Debacle

Matt
21 Oct 2004 9:53 PM
#
"Sure, right. It makes _so_ much sense to look for include files under "/Program Files/Microsoft Visual Studio 2003/vc7/PlatformSDK/include" than "/usr/include". "
Keep on making OS/Filesystem choices based on developer need. As I said earlier, in 5 years time you'll still be wondering why 97% of computer users won't touch your OS with a bargepole.
"Directory names that take a whole 10 minutes to figure out"
That could take 10 seconds to work out if they were given meaningful names. Why should I spend 10 minutes reading a man page to work out what a folder is when the developer could have made it perfectly obvious by using a long meaningful filename? If you want average users to use your OS, give the bloody folders names understandable to average people.
"["and configuration files"] Which are easily scriptable, easily transportable, and easily editable with simple tooling. "
They're not easily scriptable because there's no consistency in format. You have to make a custom script for a file with different syntax even if the task you need to perform is conceptually similar. That's called duplication of effort not ease of scripting.
"Which has done better remote access than Windows for coming up on 20 years now. Complaining about X Windows when the dominant function of most all windows systems these days is as a bitmap transport seems pretty pointless. "
You'd be hard pressed to argue X is better than RDP. Windows has caught up with remote access. X hasn't caught up in architecture. It's kludge upon kludge upon kludge. OSS advocates like to point to kludges in Windows as flaws. It's right for the other side to point to the same in X. It's hardly an example of the technically superior approach that OSS fans say it has.
"With Linux, I know I can theoretically understand and fix what's going on in my computer. "
Fine for you. Most of us don't have the time or skills to write bugfixes for our OS. It's another case of a stupid compromise. When a compromise is needed for an OS you want to become mainstream, the compromise will almost always fall in favour of the user. Certain compromises in Linux fall in favour of the developer at the expense of the user.
"The history of this industry is littered with the shells of companies that had technically superior products that got steamrollered by the competition."
You'd be hard pressed to find the decaying shell of a technically superior product that was being *given away*.
"Since that's very much tied to things like file formats and software compatibility, the obstacles to Linux adoption shouldn't be underestimated by folks on either side of the

fence"
You can't have it both ways. You can't claim that Linux (with OSS apps) is superior and does everything 90% of people need, that OpenOffice does everything that 90% of people need and opens most files that people need to migrate, and then bitch about obstacles. Linux isn't ready. The fact that its market penetration doesn't approach one tenth of what it's advocates say it could be is a very significant sign of some fundamental problem.

Stefan Kanthak
21 Oct 2004 10:02 PM
#
"Matt" (who?) wrote:

> Spaces and long filenames
> shouldn't have been an issue
> (outside of DOS/Win16/Win32
> [k?/s?/whatever?]) in 1995 let
> alone in 2004. MS has gone a fair
> way to making that so. Linux has
> not despite its proponents claims
> of technical superiority and
> superior adapatability.

Take a look at the registry of any W2K/WXP and count the entries with ~1\. Pretty, ain't it?

Install W2K/WXP on an NTFS partition and turn off 8.3 filename creation VERY early (edit the SETUPREG.HIV and/or the HIVESYS.INF), then watch how for example MS Office 2000 and later bump over filenames with spaces!

Write a (complex) batch script that can handle long filenames with spaces on it's input and pass them through some levels of FOR ..., CALL :SUB to programs.
Now do the same thing on an even ancient XENIX (MS once had this :-)!

It's sad, so sad!

Then look a bit further: Windows NT has such nice things like REG_EXPAND_SZ.
How many apps but still write those damned driver letters to their registry entries, or use C:\Program Files\ resp. C:\WINNT\ instead of %ProgramFiles% resp. %SystemRoot%?

Take a look at MSKB 249321 and 269049: both errors are SIMPLY avoidable by using %SystemRoot%\...
But don't try this with WXP: they broke it deliberately here for just THIS entry. In NT4 and W2K it worked flawlessly.

And for sure you'll even find all the lurking security flaws in
[HKEY_LOCAL_MACHINE\SOFTWARE\Classes\ChannelFile\Shell\Subscribe\Command]
@="rundll32 cdfview.dll,Subscribe %L"

I might get depressive if I'll write more ...

Mat Hall
22 Oct 2004 12:42 AM

#

"If Linux is technically superior it would have the market by now..."

Like the way that the technically superior Mac/Amiga captured the market from the PC back in the 80s? Don't be so naive -- market saturation and brand awareness are very difficult to overcome.

The whole short file name thing winds me up. It's been nearly 10 years since Win95 appeared so I find it hard to believe that anyone's still using applications that don't cope with long file names; Win 95 should have ONLY supported them for legacy (Win 3.x/DOS) apps, and then we could have probably done away with them altogether by now!

And as for "\Program Files" vs "/bin", I much prefer the latter. On my XP install I tend not to use the Program Files folder for anything if I can avoid it. I have D:\Games, D:\Apps and E:\Dev, and life is so much simpler!

Matt
22 Oct 2004 6:06 AM

#

They weren't giving Macs and Amigas away for free. We'll never know for sure but I reckon there wouldn't be a PC compatible now if they had. I don't even accept they were superior. In the early to mid 90s PC had assimilated the best technologies of both and was evolving rapidly. On the other hand Macs and Amigas seemed comparatively stagnant. (eg the A1200 was an improvement of an older design. It did not incorporate any major new features. More bitplanes, faster processor, prettier icons. Nothing astoundingly innovative though)
You might prefer /bin. You're a geek and you're familiar with it. It just doesn't make sense for average users. And there's no good technical reason that you can't use long filenames and even spaces to come up with something that is obvious and meaningful to average users. The filesystem limitations that led to those obscure names were overcome a decade or more ago. Operating systems should start taking advantage of those features.

mschaef
22 Oct 2004 6:19 AM

#

"You might prefer /bin. You're a geek and you're familiar with it. It just doesn't make sense for average users."

You still haven't bothered to explain why average users should be looking in /bin (or "\Program Files" anyway. What exactly is a user naive enough to be confused by "/bin" doing spelunking around in a directory full of binary files anyway? Windows Explorer even defaults to displaying a warning about not messing around with the files in "Program Files" unless you know what you're doing. To me, the need to post a "No Trespassing" sign explicitly speaks to the uselessness of the "user friendly" directory name.

mschaef
22 Oct 2004 6:50 AM

#

"Keep on making OS/Filesystem choices based on developer need. "

I'm not, I'm just being pragmatic about what actually matters to end users. The name of the /bin directory is not one of those things, assuming a good package manager/installer and reasonable locations for data and configuration files.

"That could take 10 seconds to work out if they were given meaningful names. Why should I spend 10 minutes reading a man page to work out what a folder is when the developer could have made it perfectly obvious by using a long meaningful filename? "

Because I'll learn the name _one_ time, type it thousands, and use it in scripts/batch files dozens. A couple minutes ahead of time to save time and increase accuracy every time I use it seems like a excellent trade off.

"You have to make a custom script for a file with different syntax even if the task you need to perform is conceptually similar. That's called duplication of effort not ease of scripting. "

You have to do the same thing with the registry. SUre, you have consistent access to keys, but given that the keys are used in radically different ways for various kinds of configuration tasks, you still have to worry about custom logic for each case.

Unix-style scripts generally allow comments (to document hidden values or the rationale behind choices), which the registry does not.

Not to mention that there's just as much black magic in knowing particular registry key names (some of which are hidden) as configuration file syntaxes.

"Fine for you. Most of us don't have the time or skills to write bugfixes for our OS. "

To be honest, I don't either.

But... if Microsoft fails to update a bug in Windows that's critical to my business, I'm screwed. With OSS, it's at least possible to look at the cost/benefits myself and hire someone to fix it (or fix it myself). In enterprise settings, where computers are sitting on hundreds of thousands of desktops or moving billions of dollars around, the ability to have that level of control over one's destiny is very valuable in and of itself. For smaller businesses, with less ability to influence MS's priorities, it can be even more important.

Speaking to my own interests: with Linux I can know for an absolute fact what my computer is doing. Not true with Microsoft: users and developers are reduced to running experiments and making guesses about how it works from what happens.

"and then bitch about obstacles."

I'm not bitching about them, I'm just stating a fact.

Legal challenges aside, OSS is about the one entity in the software business with more staying power than Microsoft, even if it fails to gain mass market adoption. In that sense, it's not a competitor that MS can out-sell, cut off its air supply, and dispatch to the graveyard. The truth is that the technical issues we're discussing don't matter worth a hill of beans: they can be fixed. The issues that do matter are all cultural: OSS represents a completely different approach to software development than Microsoft's [1]. That's how OSS will ultimately be weighed and measured.

1] I call it Microsoft's, since they pioneered for-sale software in the Microcomputer market, dating back to Bill Gates' memo to the Altair hobbyists in the mid 70's

Mat Hall
22 Oct 2004 6:56 AM
#

"...the need to post a "No Trespassing" sign explicitly speaks to the uselessness of the "user friendly" directory name."

Exactly. If you're going to warn "regular" users of the perils of playing about with things of which they do not wot, then who cares if it's called "/bin", "\Program Files", or "{CLSID-00EF-783743-746785}"? For people who do want to mess with it it's a non-issue, and giving it a short space-free name makes life easier...

It all gets a bit OT from here on in, sorry. However, I was an Amiga zealot until the mid 90s, and I still feel the pain.

"They weren't giving Macs and Amigas away for free. We'll never know for sure but I reckon there wouldn't be a PC compatible now if they had. I don't even accept they were superior. In the early to mid 90s PC had assimilated the best technologies of both and was evolving rapidly."

When did they give away free ATs? And sure, in the early 90s the PC started outstripping the Mac/Amiga in hardware terms, but only a fool would claim Win 2.x/3.x were "superior" to Workbench 2/AmigaOS. The outstripping only happened because given the choice between an Amiga ("Didn't they make the C64?") or a PC ("Ooh, we have an IBM at work!") the inferior machine won in terms of numbers purely down to the public's lack of understanding. Had the PC not carried the IBM branding for years it would never have got off the ground.

It's a sad indictment of the power of branding, but a machine with 512k of RAM (expandable up to 8Mb), a multitasking OS and GUI, graphics capabilities that made CGA look like an Etch-a-Sketch™, and real digital stereo sound lost to a machine with (typically) 64k of RAM (up to a max of 640k -- a crippled address space and no Quarterdeck EMM or DOS/4GW in those days), a command line single-tasking OS, the ability to display 4 (count them, 4!) colours at once (if you paid extra, otherwise text only), and a hideous beeping noise, and you claim that technical superiority will win? Fatuous nonsense!

Bill Gates once said that MS Word would be ported to the Amiga when it had sold a million units, but it never materialised. (It's estimated that until the demise of CBM over around 5 million of them were sold, with 1.5m in the UK alone.) With support like that, how could it hope to survive? Cluelessness amongst the public combined with powerful branding (inevitably leading to a virtual monopoly for MS) meant that there could only be one winner; once it became the defacto standard other systems had no hope but to become a niche platform...

Right, that's that settled. Next, why the Spectrum (aka Timex 2000) was the greatest computer ever... :)

Matt
22 Oct 2004 6:25 PM
#

You were indeed an Amiga zealot. It was a good machine but by the early 90s the PC had virtually caught up and in some ways surpassed it (processor performance in

particular). By the mid 90s it had entirely caught up with Win95 or WinNT.
They weren't giving away free ATs. But a previous commenter used Macs and Amigas as an analogy for Linux that they most certainly are giving away for free. Free stuff tends to beat market dominance or brand recognition. A real example is IE's slow but steady victory over Netscape (before OS integration so don't even bother moaning about that). Linux hasn't made the same gains and there's absolutely no excuse for that. There has to be some other problem. I maintain that's the choice of obscure approaches that place developer needs over user needs. Short names for the include directories, binary incompatibility, horrible packaging, nasty inconsistent configuration files are all symptoms.
And the Amiga might be around now if CBM had not neglected it. One major upgrade to the chipset in 7 or 8 years is not enough to keep in front. No wonder the PC caught up.
The Word argument is BS too. For most of the Amiga's lifetime there simply wasn't a GUI form of Word (or WordPerfect.) If developers on the Amiga had shown some initiative they might have had a decent GUI word processor before the PC. As it was they left it too late and were playing catch up with Word after that.
The reasons for the Amiga's demise isn't the fault of a stupid market that went for brand recognition over technical superiority. It's the fault of those who neglected it over nearly a decade, trying to sell the same machine in 1991/2 as they were in 1985. They gave the PC a chance to catch up and it took advantage of it.

foxyshadis
22 Oct 2004 11:57 PM
#

Not just the Amiga enthusiasts and developers, but Commodore itself basically killed it. It'd be like IBM & Intel putting out the PC, the XT, and the AT with almost no marketing, and then going bankrupt and shutting its PC division down (with no compaq or HP), leaving third-party vendors scrambling and leaving, and finally being given 386s sometime in the mid-90's. Meanwhile DEC and clones has been marketing faster, more colorful, more capacitous, more connected computers every 6-12 months with extensive 3rd party support for a decade now.

Who's going to look more attractive to customers?

You need to interoperate (nearly) seamlessly with anything the customer already uses. You need to package a full suite that does everything the customer could have done with another system, but better and more productively and even prettier. You need to _market the hell out of it_. (And not just on slashdot and blogs.) And make it almost too easy to make really cool stuff, like media players and finance reports, with python, the closest thing to a VB for linux.

Of course, do it the MS way - support exchange just dandy, but support some sendmail derivative's unique powerful _easy-to-use_ productivity features like crazy. And integrate it all - major apps built on mozilla, like explorer and html help are built on IE. You can whine about cheating, or join the big leagues who understand the needs and wants of customers. And sue a few people so you'll stay in the news and look serious about it. xD

Just saying you do things better doesn't mean squat, because you only do a few important things better (network security), but a few important things worse (file security, openoffice really needs a UI redesign, and linux as a whole desperately needs a new real GUI to rally behind), and don't do a few important things at all (for all your love of conf files, and hate of the registry and AD, those two pieces of windows are the two most important components of centralized MS management. And I refuse to believe that linux couldn't improve on them immensely if it tried. Built-in management and reporting

tools are linux's achilles heel).

Enough ranting for one night.

Raymond Chen
23 Oct 2004 5:32 PM
#

"Win 95 should have ONLY supported them for legacy (Win 3.x/DOS) apps..."

But Windows 95 needed to be compatible with Windows NT, which did support SFNs for 32-bit apps.

The problem with saying "32-bit apps cannot see SFNs" is that you create interop problems at the boundaries. A 16-bit app launches "notepad SHORTN~1.TXT" and Notepad (a 32-bit app) puts up the error message "File not found".

DrPizza: If you shim litware.exe's command line parser then you risk breaking cases where litware expects to receive a long name by means other than the file system association. Perhaps litware didn't mind, but this change was made during a service pack which has much shorter testing cycles.

Norman Diamond
24 Oct 2004 8:57 PM
#

10/20/2004 11:02 PM Raymond Chen

>> "Does Windows Explorer support long
>> filenames?"
>
> I'm sure this is some sort of trick
> question.

You're right. I had intended it as simple acrimony rather than trickery, but after thinking about it, you're right that it does require trickery to interpret it.

The literal answer is sometimes yes and sometimes no, but the literal answer provides no useful information.

The trickery involves thinking about what caused the question to arise in the first place, and why the answer isn't a consistent yes.

The full answer requires observing one of the ways a user might use Windows Explorer, in this case making backups onto a USB hard drive. In Windows 2000, and I think 98 and ME, when a USB hard drive is attached, Windows Explorer recognizes the added drive and can write long filenames to it. In Windows XP this was broken until SP2, now SP2 can write long filenames to it. In Windows Server 2003 it remains broken either with or without SP1 beta build 1218. In broken versions of XP and 2003, you can even look at long filenames already existing on the USB hard drive, try to cut from one directory and paste into a different directory, and watch Windows Explorer say that it can't write long filenames and offer a truncated 8.3 filename.

Raymond Chen
24 Oct 2004 9:04 PM
#

Sounds like an issue with the filesystem then. Explorer uses the GetVolumeInformation function to determine whether a drive supports long file names. If the maximum component length is 12 (8+dot+3), then Explorer says "Sorry, this is not an LFN drive."

Explorer is the victim here.

PS, could you tone down the nastiness? You could have said, "Sometimes Explorer thinks my USB drive doesn't support long file names" instead of saying "Explorer doesn't support long file names, that piece of crap".

DrPizza
25 Oct 2004 9:51 AM
#

"In Windows Server 2003 it remains broken either with or without SP1 beta build 1218. " News to me.

I have LFNs aplenty on my FAT USB hard drives.

What the hell are you talking about?

Norman Diamond
25 Oct 2004 5:58 PM
#

10/24/2004 9:04 PM Raymond Chen

> Sounds like an issue with the filesystem
> then. Explorer uses the GetVolumeInformation
> function to determine whether a drive
> supports long file names.

Does GetVolumeInformation change its answer from one call to the next? Does it change its answer depending on which process id has called it? Different instances of Windows Explorer don't always agree on whether a long file name can't be moved from one folder to another, they only sometimes agree on it.

10/25/2004 9:51 AM DrPizza

> I have LFNs aplenty on my FAT USB hard
> drives.

So do I. Windows 2000 can write them, Windows XP SP2 can write them, and depending on factors yet unknown, Windows XP prior to SP2 and Windows Server 2003 can sometimes write them. As mentioned, some of the occasions when Windows Explorer denies ability to write them and offers to truncate them to 8.3 filenames are when doing cut-and-paste from one folder to another on the same drive, so I assure you I've got them there.

There's a bit too much randomness involved in the unknown factor(s), but here's a combination that reproduces it somewhat comparatively reliably (in XP pre-SP2 and in

2003). After booting and logging in, open a Windows Explorer window, unfold "my computer" and go to some directory on your internal drive where perhaps you have a file that you want to make a backup of, and keep the window open. Attach a USB hard drive that is "self-powered" (i.e. needing its own AC adapter, being powered by its own adapter instead of from the PC's USB circuitry). Sometimes an icon doesn't appear at all in the left-hand pane of Windows Explorer but you can type the drive letter and a colon into the address bar and Windows Explorer will move its view to it. Sometimes a drive icon does appear so you can just click it and Windows Explorer will move the way it should. You can see your exsiting LFNs. But when you try copying or moving an LFN to the drive (either from your internal drive or from a different folder on the same external drive), Windows Explorer refuses.

If you open Computer Management, go to Disk Management, right-click the partition displayed for the external drive, and select "explore", then a new Windows Explorer window opens that sees the external drive. That Explorer window is usually pretty reliable -- until you click the Refresh button for whatever reason.

Michael
25 Oct 2004 6:22 PM
#

"If Linux is technically superior it would have the market by now (ten years after widespread distribution). "

If people were not brainwashed morons, they would not buy bottled water thinking that they are getting cleaner and more pure water than from the tap. Ever played Lemmings?

About SystemRoot, etc. You guys should not forget that english is not the only language Windows supports. There are quite a few Windows versions which use different names for Program Files even in Latin charset. There are other charsets too. For example, the encoding for russian DOS and russian Windows is different. I don't know about NT/Win2K, but in Win3.x/9x short filenames are stored in cp866 on disk to support DOS programs, and converted to codepage 1251 when "Open File" dialog is opened by Windows program. LFN are encoded in Unicode, so using LFN is kind of more uniform. I think there is no conversion to open a file with LFN. And be sure, that Program Files not only has a different name in russian Windows, it is also spelled in cyrillic.

I think Raymond can tell more fun things about codepage conversions.

David Candy
27 Oct 2004 3:27 PM
#

Chris Becke,

This is a list from http://users.lia.net/chris/win32/explorer.asp

Some look to be cmd specific (such as %~n1 which extracts the name). So pass it to a bat and

program.exe %~n1

or for path and name

program %~dpn1 (dpn is drive, path, name. x is extension see the help on For)

%h and %s are used mainly in DDE conversation and I've never seen &h used at all. Explorer uses %s to pass the window state of the calling window (eg c:\) when opening a sub folder. If you replace it with a showwindow const then all windows become that (eh maximised). IE has quite a few parameters but none seem to be the show state. %I is only used by Explorer and is two numbers (at least if you pass it to a bat file to echo %1) :nnnn:nnnn (in decimal) with the second being the folder and the first the number of the item in the folder. They exist only while the folder is opened. ? is for pif files only.

%h, when creating a window you can assign a hotkey to the window. Presumbably this is for passing hotkeys from window to window. Though I've never see any program use this feature.

Now for that list

%% - escape sequence for '%'
%0 - the thing being clicked (or dropped upon)
%1 - represents the name of the item being clicked (or dropped upon).
%2 - 1st file "dropped" onto item. (in case of files being dropped).
%3 - 2nd file "dropped" onto item. etc...
%* - all the dropped items.
%L - same as %1 (unconditional lfn form?)
%I - generates a global itemid of the form ":-2109146316" (folders only?)
? - Will prompt the user for the commandline to supply

I don't know what these do, but they do something:

%L = document name
%0 = short version of document name (sometimes)
%1 = same as %0
%2 = parameter 1
%9 = parameter 8
%~2 = parameter 1 onwards
%~9 = parameter 8 onwards
%* = all parameters (i.e. same as %~2)
%H = hotkey in hex. can only be used once
%I = pidl (different format depending on OS)
%S = initial Show format in decimal

AND RAYMOND ANSWERED ONE OF MY QUESTIONS !! (and he says my questions aren't interesting to anyone but this got 60 responses, most mini essays).

David Candy
27 Oct 2004 3:34 PM
#
>Observe that the spaces in the path "C:\Program Files\Litware Deluxe\litware.exe" are not quoted in the program registration. Consequently, the shell mistakenly believes that the program name is "C:\Program", which it cannot find. The shell therefore plays it safe and assumes no LFN support.

I was going to ask this last week - how can the shell find the program to pass it a short file name then?

I just paraphrase what I think was said

If the shell can't find the program file it passes a short name to the program (that it can't find) and everything works. As Pauline Hanson would say "Please Explain".

Raymond Chen
27 Oct 2004 3:39 PM
#
Because the shell hands the final command line to CreateProcess - CreateProcess has its own search algorithm.

David Candy
28 Oct 2004 12:44 AM
#
So that's why AppPath entries are ignored in Registry command strings.

HKLM\...\AppPaths
prog.exe=c:\somewhere\prog.exe

HKCR\.ext\shell\open\command
@=prog %1

won't work.

Matt
30 Oct 2004 7:31 AM
#
"""
"If Linux is technically superior it would have the market by now (ten years after widespread distribution). "
If people were not brainwashed morons, they would not buy bottled water thinking that they are getting cleaner and more pure water than from the tap. Ever played Lemmings?
"""
Lovely. People are using Windows over Linux because they are brainwashed morons. Again (and again and again and again) that's the attitude that keeps people away from Linux.
Run Windows? Moron! Can't get modem/video/network card to work under Linux? Moron! Can't recompile kernel to support obscure feature for funny application? Moron! Don't know the ins and outs of Unix filesystems and partitioning? Moron! Don't understand three letter directory names? Moron!
And you people will still be surprised in 5 years time when the market share of Linux is no better than it is today!

Stefan Kanthak
30 Oct 2004 7:22 PM
#
Raymond Chen wrote:

> Because the shell hands the final

> command line to CreateProcess -
> CreateProcess has its own search
> algorithm.

Hmmm... does this mean:
a) the shell does no search at all?
b) has another search algorithm than CreateProcess()? Which?
Can you say how simple/relative filenames are handled/interpreted?
a) relative to the CWD?
b) ....?

Can you say a word or two why filenames in the registry are
a) not always fully qualified?
b) not always specified fully qualified as REG_EXPAND_SZ?

Raymond Chen
30 Oct 2004 9:17 PM
#
(b) the shell implements its own search algorithm. Just fully quality (and quote) your paths and the precise algorithm becomes irrelevant.

Why aren't file names in the registry fully qualified/REG_EXPAND_SZ? You'll have to ask the programs who wrote those registry keys.

Stefan Kanthak
31 Oct 2004 6:46 PM
#
Hi Raymond,

you wrote:

> (b) the shell implements its own
> search algorithm. Just fully
> quality (and quote) your paths and
> the precise algorithm becomes irrelevant.

I always specify my entries with fully qualified and quoted path and use REG_EXPAND_SZ whereever possible, but Windows' shell itself doesn't: I count 2 references to explorer (yuck, even without extension) and 8 to explorer.exe in the registry of Windows 2000. rundll32 has 8 references without path and extension and about 20 without path. And both explorer.exe and rundll32.exe are not the only programs referenced without path!

I count NUMEROUS references to ole32.dll, shell32.dll, shdocvw.dll and other dll's, ocx's, ... without path in registry entries for "InProcHandler32", "LocalServer", "LocalServer32" and "InProcServer32" and I don't feel comfortable about it knowing that PATH always includes . at first component: as long as SafeDLLSearchMode ain't set (before Windows 2003!) this invites for exploits!
Cf. 249321 and 269049

That's what scares/bothers me!
"trustworthy computing" anyone?

Raymond Chen
31 Oct 2004 10:47 PM
#
The fashion for fully-qualifying paths for files that normally reside in the system32 directory has gone through its own cycles. For a while, using a relative path was recommended to allow for side-by-side application overrides. I don't know whether that's still the fashion or not.

David Candy
1 Nov 2004 1:11 PM
#
The shell algorithm is good to know for diagnosing problems with systems and file associations.

Chris Becke
1 Nov 2004 11:59 PM
#
Ha. Thats ironic. Having my own (ancient) web page thrown back at me :)

As always, I desire official sources of information I... really don't know how I got.