The Old New Thing

Why doesn't Explorer show recursive directory size as an optional column?

29 Oct 2007 10:00 AM | 43

"Why start up another program to see folder sizes, when they should just be right there, in Explorer, all the time?"

The same reason \\ does not autocomplete to all the computers on the network: Because it would destroy corporate networks.

Showing folder sizes "all the time" means that when you open, say, the root of a large server, Explorer would start running around recursively enumerating every single directory on the server in order to compute the folder sizes. One person doing this to a server is bad enough. Imagine if hundreds of people did it simultaneously: The server would be hammered continously.

Even worse: imagine doing this across a limited-bandwidth link like a VPN or an overseas link. The link would be saturated with file enumerations and wouldn't have any bandwidth remaining for "real work". Even the change-notifications that Explorer registers are cause for much hair-pulling on corporate networks. (And these are change-notifications, which are passive.)

Even on a home computer, computing folder sizes automatically is is still not a good idea. How would you like it if opening a folder caused Explorer to start churning your disk computing all the folder sizes recursively? (Then again, maybe you don't mind, in which case, go nuts.)

(Of course, the question sidesteps the question the linked article tries to address, namely, "What do you mean by the size of a directory anyway?")

Blog - Comment List MSDN TechNet

Comments

xix
29 Oct 2007 10:30 AM
#

Given the fury and gnashing of teeth I endure when getting the properties of a folder to obtain this bit of information, I can't imagine how utterly unresponsive my PC would be if it did this for all folders, all the time (or even some of it).

One might think the recursive size could go into folder metadata and updated only when contents changed (recursively). That would probably involve a lot of gnashing as well, and several time machines, alas.

Though I rarely seem to need this information anyway, thankfully.

ac
29 Oct 2007 10:56 AM
#

One of the shell exstensions I always install is Folder Size (Property page tab, not column)

http://www.pcworld.com/downloads/file/fid,15304-order,1-page,1-c,alldownloads/description.html

http://www.softcom.net/users/mikey719/foldersize_screenshot.html

(Author homepage seems to be gone)

Name required
29 Oct 2007 11:16 AM
#

Control Panel (or at least, things accessible from there) can tell which disks are local HDs and which are not, as can Explorer (how else could it offer to allow you to search "Local Hard Drives (C:; D:). But it is too difficult to do that to have this feature for local HDs only?

I'm not saying the feature is a good idea - it is something one generally only needs to know pretty rarely - but saying it is because clients would take the network down as a result of it being present is a little strange.

Scott
29 Oct 2007 11:24 AM
#

It always seemed to me that the file system could store this as metadata and keep it up to date as files changed. That way, it wouldn't have to enumerate all the files to get the size.

Scott

nksingh
29 Oct 2007 12:02 PM
#

The feature probably wouldn't interact well with symbolic links and the various types of NTFS reparse points. I would have liked this feature as well, though.

It would be nice if this information could go into one of the attributes of a directory MFT record, but it would be tough to know what to do if it ever gets out of sync... how do you know if the value you see there is accurate and when do you just have to recalculate it from scratch?

sean
29 Oct 2007 12:12 PM
#

If you only need to know recursive size rarely, just use explorer. Right-click on the directory, select properties. See size, size on disk along with sub-directory and file counts.

640k
29 Oct 2007 12:23 PM
#

The tooltip shown in explorer do exactly this.

1. The tooltip destroys corporate networks/bandwith limited environments

2. The tooltip shows a ambiguous size

Why?

[*Your second statement answers the first. -Raymond*]

James Schend
29 Oct 2007 12:23 PM
#

Coming from a Mac Classic background, I always install the Folder Size column first thing. (Well, I haven't in Vista yet, waiting until I know for sure it's compatible and stable). I guess Apple just doesn't care about killing corporate networks, hah.

Leo Davidson
29 Oct 2007 12:24 PM
#

There is a good way to do this though, and it's what the file manager I use does. (I'll avoid dropping the name but see my homepage for some stuff I've written about the program if you are interested.)

The user can click a "Get Sizes" button on the toolbar which populates the normal size column for folders. If folders are selected when the button is pushed then only their sizes are calcualted; else all of them are calculated.

When the user wants the information it's a single button click away, and doesn't even take up the space of an extra column. When they aren't interested it isn't needlessly calculated.

This is much better than using the Properties dialog because it lets you see, sort and compare multiple folders at once. You can see the combined size of arbitrary combinations of folders by selecting them and checking the status bar.

Having become used to this I wouldn't want to be without it now.

(The same program also lets you turn on automatic calculation in specific directories if you want to. I've never used that but I guess it would be useful in places where you know the directories are shallow, not on a network, and often need to see their sizes.)

Evan
29 Oct 2007 12:44 PM
#

Here's a question on the proper semantics:

If a user has a partition mounted at c:\ and another at c:\foo\bar, should the folder size for c:\foo include the bar subdirectory?

Any tool that does this has to make a decision on this aspect, and I'm not convinced there is a single right answer. On one hand, if you're using it to clear up space, the answer should be no: freeing up c:\foo\bar won't increase the space available in c:\. On the other hand, *not* including it seems wrong -- bar is, after all, inside foo.

Similar issues -- perhaps thornier -- also arise with junction points and such.

Gabe
29 Oct 2007 12:45 PM
#

James Schend: It was actually much worse. When you had a Finder window open to a network folder, it would poll for changes -- constantly. This was especially bad on LocalTalk, which ran at only 230kbps. If somebody noticed that the network (or just a computer) was running slow, there was a good chance that closing all the Finder windows would fix it.

Evan
29 Oct 2007 12:47 PM
#

Also, I'm partial to the program WinDirStat for this. (http://windirstat.info/)

Shows you nice things like a graphical indication of directory sizes, so you can spot at a glance the big wasters.

Ma
29 Oct 2007 12:55 PM
#

Compressed / uncompressed sizes?

Permissions?

Patrick Farrell
29 Oct 2007 1:09 PM
#

Local (and possibly network) directory sizes could be shown all the time if it were trivial to query that data. In my mind, recursive discovery of this information is an "old school" way of thinking.

Apple and ReiserFS have blazed the way into new capabilities that could allow this to be a O(1) complexity situation. Well, maybe O(n). By updating a database at the kernel instruction level, you could have make this data as easy to discover as a database query.

It was my understanding that WinFS would help push us towards this type of functionality, but as we've seen with WinFS and ResierFS, this type of functionality is really difficult to acheive.

Evan
29 Oct 2007 1:26 PM
#

"Apple and ReiserFS have blazed the way into new capabilities that could allow this to be a O(1) complexity situation. Well, maybe O(n)."

O(n) what? Doing a full traversal to find the size of a directory is O(n) in probably the number of files, which is is the most natural thing for n to be. [The problem then is that n is exponential in the depth of the directory tree.]

And how do they do this without storing the size of each directory in the directory node itself, and updating it on each save? If that's how they do it, unless they use that data otherwise I don't think it's worth it. (I'm interested in FS design a bit, so I'm genuinely interested in the answer.)

MS
29 Oct 2007 1:49 PM
#

"It always seemed to me that the file system could store this as metadata and keep it up to date as files changed."

That might seem to work, but the extra overhead to every write operation would add up (I would imagine a program like a database would do lots of writes in a single folder in a short time frame). Also, there are probably quite a few different ways to modify files on disk; some of which are not conducive to being "tracked." I could write a program that did not use a single Windows API that tracked what I was doing to a folder's contents.

Katla
29 Oct 2007 2:50 PM

#

We use a program called FolderSizes for this type of thing (http://www.foldersizes.com). It does lots of other stuff, too, but I can right-click in Explorer and launch it for a quick view.

I absolutely prefer having this type of capability as a separate application, rather than integrated into the shell. It's excruciating enough waiting for Windows to surmise the size of installed apps within Add/Remove Programs. :-)

Reinder Verlinde
29 Oct 2007 3:27 PM
#

"Showing folder sizes "all the time" means that when you open, say, the root of a large server, Explorer would start running around recursively enumerating every single directory on the server in order to compute the folder sizes.

One person doing this to a server is bad enough. Imagine if hundreds of people did it simultaneously: The server would be hammered continously."

That argument reminds me of an argument in the command-line versus GUI discussions of the 80s: "why would you go and fetch an entire directory listing? Just let the user type the name of the file he wants; showing tens of files wastes precious cycles."

Just like then, I think the answer should be "because a) we can, and b) it makes things easier for the user".

I think one can implement this reasonably efficiently, as follows:

1. To decrease network traffic, the server should do the entire enumeration.

2. The server should have something like FSEvents in Mac OS X (basically an efficient way to ask "which directories changed on this disk since the last time I asked?". That would allow it to bring its (directory ID => total Size) cache up to date whenever a "how large is this directory?" request comes in. If the cache is up-to-date, a lookup would be cheap on disk I/O.

3. Callers of the "how large is this directory?" API should anticipate the case where the answer comes late or not at all. This allows the server to postpone answering or even ignore these questions if it thinks it has better things to do (on the Mac, the Finder has never blocked waiting for answers here; it just shows "--" as size until it knows a better number)

4. For hard links, compression, etc, the implementers will have to make some potentially tough choices (my guess would be to keep three sizes: sum of uncompressed file sizes, sum of bytes stored, #bytes on disk)

[*Ooh, changing a network protocol. That comes with its own massive problems. - Raymond*]

dan drake
29 Oct 2007 3:46 PM

#

I believe it does recurse directories. You just have to ask it to.

1.) Go to a network share. \\servername\sharename

2.) pull up the context menu by holding down the right <most of the time> mouse button on a folder

3.) select properties

if you don't believe it, run wireshark to get a network trace.

Kuwanger
29 Oct 2007 3:50 PM
#

I really don't understand the argument. Isn't one of the main ideas behind corporate networks is that you can lock down settings? So, why couldn't you just disable the "shown recursive directory size as an optional column"? (Feel free to point out how naive this comment is.)

DriverDude
29 Oct 2007 5:11 PM
#

"(on the Mac, the Finder has never blocked waiting for answers here; it just shows "--" as size until it knows a better number)"

Neither does Explorer's Property pages. But the confusing part is it doesn't indicate when it is done recursing directories. It counts up and you just have to look at it until it stops counting.

Brio
29 Oct 2007 5:19 PM
#

Hey, I'm the developer of the Folder Size program that Raymond linked to. As a long time fan of The Old New Thing, I'm honoured! :)

Here are some of the issues I dealt with making this utility.

First of all, I completely agree with Reinder's post, that this stuff really should be possible with today's computers. Getting it to perform well is an engineering challenge, not a reason to avoid the issue.

xix, you assume that this functionality would make your PC unresponsive while it was adding up all the sizes. Early versions of Folder Size did slow disk access quite a bit, so the current version monitors the length of the disk request queue, and halts its background scans while other programs are using the disk. Since I implemented this, I haven't noticed any significant performance degradation. You also command that you rarely seem to need the information, but I would argue that when the information is always available, you tend to rely on it more!

rksingh, you are right, NTFS reparse points are tricky, so Folder Size just ignores those for now. Following the links blindly is asking for trouble (infinite cycles!) Ideally it could follow the links intelligently and recognize where the links are going, but that just hasn't been implemented.

Ma, you are right - permissions are an issue. Folder Size scans local drives as the SYSTEM user, which typically has access to all files. This is a security problem, because users can see folder size totals about folders they don't have access to (which could, for example, reveal your hidden porn collection!). For network drives, they are scanned as the calling user, so there is no security issue there.

Folder Size currently reports files the way Explorer does (which makes Folder Size a suitable replacement for Explorer's Size). So, uncompressed sizes are reported, and alternate NTFS streams are ignored. I think the correct solution would involve 2 columns, logical size (what it currently does) and physical size (which would also count unused cluster space). I haven't found the difference to be all that important.

The other big issue is Vista. Folder Size does not work at all in Vista now, because Vista completely removes the IColumnProvider shell extension API! It is replaced with a new Property system. But the property system doesn't seem quite right for this functionality.

I'm currently working on an update that will allow disabling the scanning for different drive types, so that may make the program more useful for larger networks. Right now it's all or nothing!

So yeah, there are some rough edges, but for me, this program is the best solution available for this problem.

And the source code is right there on the web, so if anyone wants to fix any of these issues or add Vista support, please submit a patch!

Brio
29 Oct 2007 5:26 PM
#

DriverDude: You hit on areas where Folder Size is better than the Mac's implementation. On the Mac, it just scans your folders one at a time, displaying -- until it's done, and completing each folder before going to the next. Folder Size scans them simultaneously (round robin) so you never get stuck on a big folder if you actually want to see the size of a small folder further down the list. Also, Folder Size shows a + symbol while scanning, so you know when it is done.

Thorsten
29 Oct 2007 5:47 PM
#

Brio: I haven't looked at the source, but I assume that when accessing network folders the "local" folder size service is doing the counting and caching of folder sizes. A solution to the "it would destroy corporate networks" issue might be that the folder size service for network drives first tries to contact the folder size service that might be running on that remote computer and if there is a folder size service running just forwards the requests. In that case the folder size service on the server would do the caching and the traffic over the network is limited to the requests between the 2 services.

Btw, since I discovered Folder Size a couple of years ago it's always pretty much the first thing I install after the OS. Thanks!

yonilevy
29 Oct 2007 6:07 PM
#

The real question is whether the "folder size" will be displayed in explorer once WinFS is available. Since it does indexing, this information could be obtained with no cost at all (unless I'm missing something).

I too am a big fan of the foldersize shell extension, especially when I'm trying to free disk space.

Kuwanger
29 Oct 2007 6:13 PM
#

@Brio: "permissions are an issue. Folder Size scans local drives as the SYSTEM user, which typically has access to all files. This is a security problem, because users can see folder size totals about folders they don't have access to (which could, for example, reveal your hidden porn collection!)."

A stupid question, but why doesn't Folder Size just scan local drives under the user's account? The only place I can see where it'd make any sense to elevate to SYSTEM user would be if the user can trivially become a SYSTEM user; but, even then, it'd seem appropriate for a setting on whether to elevate to SYSTEM user.

Maurits [MSFT]
29 Oct 2007 6:32 PM
#

+1 on the idea of baking folder size into the file system as update-on-write metadata. Yes, you lose a level of normalization. No, that's not a problem. No, compressed/link/mount aren't a problem either.

Evan
29 Oct 2007 6:55 PM
#

"+1 on the idea of baking folder size into the file system as update-on-write metadata. Yes, you lose a level of normalization. No, that's not a problem. No, compressed/link/mount aren't a problem either."

Yes, you'll have to do several more writes per update to propagate those changes up. (Modulo caching and such.) Maybe, that might affect performance in a way that matters. Probably, you'd want it configurable because of that, much like the noatime option. Yes, that adds more complexity.

Brio
29 Oct 2007 7:38 PM
#

Kuwanger: No you're not missing anything, the only reason that wasn't done is speed. If the scanner used the user's account, it would need to store separate data structures for each users, which would increase memory usage dramatically, and require an extra change notify handle for each user. Or, the internal data structure could store user permissions for every folder, and for every file access look up its user permission info... which just got too complex for my brain. :)

Seemed to be a lot of work and/or performance penalty for something few people would notice.

Thorsten: Yes, that is a good idea for networked machines. Of course, it still has the permissions problem I just discussed, which is likely a more serious security problem for networked drives.

With the current implementation, permissions are respected for network shares, but violated for local drives.

In most cases, I implemented the features that I personally wanted, and then had little motivation to go further. I have a day job. :)

I'm glad the people who find it useful do, and I'm open to someone who wants to fix some of these known limitations!

Merus
29 Oct 2007 8:12 PM
#

I have a program that scans my computer, computing the sizes of every directory and then showing the relative sizes of each file in a chart.

This process takes several minutes. The average modern computer contains somewhere in the order of 100,000 files.

I can understand why it's not the default Windows behaviour.

Brio
29 Oct 2007 8:23 PM
#

Merus: That's exactly why it DOESN'T make sense to do this scan in a separate program!

When you want to see the information, you have to wait there for several minutes! Why not scan during your hard drive's "idle cycles"?

Miral
29 Oct 2007 8:59 PM
#

Whenever I want to know the sizes of things, it's usually for one of two reasons:

1. I want to copy something onto a CD/DVD/USB or other limited storage device

2. I'm running out of disk space

For #1, Explorer's properties window is sufficient since I can see the collective size of everything I'm wanting to put somewhere else.

For #2, I use a program called SequoiaView, since that graphically represents everything (so the biggest files/folders take up the largest blocks on the screen), which is a great help. (Although for some reason the "XP" version doesn't actually work on XP. You need to use the regular version on all versions of Windows.)

::Wendy::
29 Oct 2007 10:31 PM
#

that sounds like a good name for an anarchist pop all-girl band. Who needs the spice girls when you've got:

"Why doesn't Explorer show recursive directory size as an optional column?"

or

WDESRDSAAOC for short

Cheong
29 Oct 2007 11:23 PM
#

It does seems to be nice to have it show the folder size (including all files and subdirectories) if it can drain data out of the indexing service. (I have no idea whether indexing service also saves size data, though.)

Igor
29 Oct 2007 11:38 PM
#

I disagree that storing the folder size as metadata would add to the overhead.

You are already updating directory node when its size is changing.

If you just finished writing a new file, you will have to write last access time for it to a directory node so why not write the new size at the same time?

If you just deleted a file you are removing it from directory node and it is easy to subtract its size.

Even propagating this information upwards should not present too much of a problem or performance hit. If the size is constantly changing (like when you are downloading a file) without using file manager), it could be cached because it is not vital information and written out only after file updating is finished.

In my opinion, current filesystems lack an option to pass the final file size on file creation so the fs driver can allocate continuous storage and avoid fragmentation -- it would also help in this case because once created, file size wouldn't change. Some filemanagers and p2p applications already implement this to avoid fragmentation, too bad they don't have system support.

Worf
30 Oct 2007 1:48 AM
#

It's more fun if you do embedded development - the Microsoft adaptation kits for Windows CE, and Windows Mobile can easily have 80k+ files (each) in it. Takes a good 10 minutes for the property page to finish.

Also a great way to stress test NTFS - defragging's good until you do a clean build that turns the sea of blue into red (the build can touch a good portion of those files).

Now imagine having Folder Size listed with a build churning the disk in 10 explorer windows...

Gabe
30 Oct 2007 3:11 AM
#

Igor: Windows most definitely passes the initial allocation size to filesystems. It is passed to NtCreateFile as the AllocationSize parameter, which one can presume is passed to the filesystem in the Irp->Overlay.AllocationSize field of the IRP_MJ_CREATE request.

If you were going to implement this feature in the filesystem metadata, you would probably want to store the allocation size. That eliminates the problems associated with having to traverse links (cycles, network shares, etc.), but still brings the question of how to add hard links. It also means that it would only require a directory's metadata to be updated when a cluster is allocated or deallocated, which is likely much less often than when a file's nominal size changes. Since space (de)allocation requires writing all over the place anyway, it might not be much overhead to update the directory's metadata at the same time.

The problem is that this would still require changes to be made all the way up to the root. In other words, maintaining this data would be O(M*N) where M is the number of allocation changes and N is the average height of the directory tree. Caching is not an option because metadata is logged, so you couldn't just save up directory size changes until there was a convenient time to write them to disk.

As an added bonus, though, keeping this data would make it trivial to enforce directory size quotas.

A different option to store the data persistently is to have the indexing engine maintain it. It wouldn't be real-time and you would only get it for parts of the drive that are indexed, but it's certainly less overhead than recursively recomputing it all the time.

As a point of reference, my 80GB laptop hard drive has 350,000 objects, 10% of which are directories, and it takes SequoiaView over 8 minutes to traverse its tree. The tree is 16 deep, with the average file being 7 directories deep.

Dewi Morgan
30 Oct 2007 3:13 AM
#

While storing the size of the "contained stuff" seems reasonable in a folder, it could only store the size of the "immediately contained stuff - storing metadata about subfolders is complex without polling.

A possible proposal (at least within a single filesystem) is for folders to store everything, initially: the size of their immediate contents, and the and the size of everything they contain (initially both are 0, which is nice).

If they contain something from outside the filesystem, a reference to that thing is passed to each element between there and the tree root.

If a file is modified, a reference to that thing is passed to each element between there and the tree root, and its previous size is subtracted from their sizes.

Anytime a modified file gets its size queried, it stores the size again and the reference is removed.

Worst possible additional storage for this, where files average a directory depth of n, is equal to n copies of the FAT.

Querying any item that didn't know its size would involve looking up the stored size number and then checking the size of any "modified or external" references, returning the value, then updating their sizes in all their parents. Speed then scales with the number of recently-modified things, rather than the number of things.

Modifying an item is slowed by writing to all parent folders - even put as an "uber-low-priority" write, this could still be a problem.

I don't think something like that is worth doing: treesize queries seem too uncommon to optimise for. Not worth the tradeoffs.

Patrick Farrell
30 Oct 2007 9:57 AM
#

@Evan: "O(n) what?"

Thanks for calling me out on that. What I was indicating with O(1) was there would be a database call for the directory to retrieve the metadata for each of the child nodes.

What I meant by O(n) was there would be a call for each node to determine metadata (which might be more efficient depending on how the caller pages the data).

I will not try to act like I'm an expert on the details, but I do know that Spotlight, from Apple, updates an internal metadata store as files are altered at the processor level. Watching it work from my Windows vantage was nothing short of miraculous.

I am thinking along the lines of how Indexing service can provide highly efficient searches because it avoids the computational (and network overhead) associated with discovery of the data (because it's already cached). Apple does this without the overhead (and batch oriented nature) of the Indexing service. It accomplishes this by updating the internal metadatabase as crud operations are performed on files.

Igor
30 Oct 2007 10:48 AM
#

Gabe said : "Windows most definitely passes the initial allocation size to filesystems."

Perhaps NtCreateFile, but not CreateFile(), nor fopen(), lopen(), creat() or any other API programmers usually use.

@Dewi Morgan:

How about each folder storing the size of its own files not counting subfolders and junctions?

That way you don't need to propagate info upwards through the filesystem (because changing subfolder size would not affect parent folder) and the part of the OS which implements size display then just adds them up on as needed basis. This could be cached and lazy-updated.

Maurits [MSFT]
30 Oct 2007 4:51 PM
#

> Yes, you'll have to do several more writes per update to propagate those changes up.

Indeed. But this isn't a problem either. Writes, in typical usage scenarios, are much less frequent than reads.

Now, if you were running on a server, and you needed to write frequently to log files that were nested 300 folders deep, this might affect performance. So, like all new features, there should be a way to turn this off.

But it's (usually) a heck of a lot faster than calculating folder size the hard way.

Igor
30 Oct 2007 8:59 PM
#

No need to propagate at all.

余啊雷
4 Mar 2009 10:27 AM
#

PingBack from http://blogs.msdn.com/oldnewthing/archive/2009/02/17/9426787.aspx