The Old New Thing

Why doesn't the file system have a function that tells you the number of files in a directory?

17 Feb 2009 10:00 AM | 46

There are any number of bits of information you might want to query from the file system, such as the number of files in a directory or the total size of the files in a directory. Why doesn't the file system keep track of these things?

Well, of course, one answer is that it certainly couldn't keep track of every possible fragment of information anybody could possibly want, because that would be an infinite amount of information. But another reason is simply a restatement of the principle we learned last time: Because the file system doesn't keep track of information it doesn't need.

The file system doesn't care how many files there are in the directory. It also doesn't care how many bytes of disk space are consumed by the files in the directory (and its subdirectories). Since it doesn't care, it doesn't bother maintaining that information, and consequently it avoids all the annoying problems that come with attempting to maintain the information.

For example, one thing I noticed about many of the proposals for maintaining the size of a directory in the file system is that very few of them addressed the issue of hard links. Suppose a directory contains two hard links to the same underlying file. Should that file be double-counted? If a file has 200 hard links, then a change to the size of the file would require updating the size field in 200 directories, not just one as one commenter postulated. (Besides, the file size isn't kept in the directory entry anyway.)

Another issue most people ignored was security. If you're going to keep track of the recursive directory size, you have to make sure to return values consistent with *each user's* permissions. If a user does not have permission to see the files in a particular directory, you'd better not include the sizes of those files in the "recursive directory size" value when that user goes asking for it. That would be an information disclosure security vulnerability. Now all of a sudden that single 64-bit value is now a complicated set of values, each with a different ACL that controls which users each value applies to. And if you change the ACL on a file, the file system would have to update the file sizes for each of the directories that contains the file, because the change in ACL may result in a file becoming visible to one user and invisible to another.

Yet another cost many people failed to take into account is just the amount of disk I/O, particular writes, that would be required. Generating additional write I/O is a bad idea in general, particularly on media with a limited number of write cycles like USB thumb drives. One commenter did note that this metadata could not be lazy-written because a poorly-timed power outage would result in the cached value being out of sync with the actual value.

Indeed the added cost of all the metadata writes is one of the reasons why Windows Vista no longer updates the Last Access time by default.

Bonus chatter: My colleague Aaron Margosis points out a related topic over on the ntdebugging blog: *NTFS Misreports Free Space?* on the difficulties of accurate accounting, especially in the face of permissions which don't grant you total access to the drive.

Blog - Comment List MSDN TechNet

Comments

DWalker
17 Feb 2009 11:09 AM
#

Showing the size of files that the user doesn't have access to is an "information security disclosure vulnerability"?

I suppose so, in the strictest sense, but since I am the OWNER of my computer, and all of the files belong to me -- and I could take ownership of any files that didn't belong to me -- I use one of the free services that maintains Folder Size information. I don't really know or care what it does for files that I don't have permission to (such as OS files). I have read permission for the bulk of the OS files anyway. It occasionally takes a few seconds to refresh all of the numbers, but that's fine with me.

I can see how much space is used recursively by all of the files in a subfolder, and how many files are present. The information might not be 100% accurate in the case of hard links, but I don't really care, since I don't have lots of hard links (that I know of). I feel that the usefulness of the (approximate) information outweighs the potential small edge cases where some piece of it is not 100% accurate.

Maybe Microsoft can't get by with something that's not 100% accurate, and I don't begrudge them not implementing this feature, but what I use is terrific and does a fantastic job.

[*And you don't mind exposing that information to non-OWNER users? -Raymond*]

DWalker
17 Feb 2009 11:32 AM
#

My computer doesn't HAVE any non-owner users. Ever. But no, in general, I wouldn't mind at all.

I suppose the feature should be turn-off-able for those who DID mind showing this info.

Michiel
17 Feb 2009 11:35 AM
#

Considering the OWNER user is most likely to need the information (as he's likely the maintainer of the sytem), the valid question is "Do I mind NOT exposing that information to non-OWNERs", and the answer is "No, they can use the old mechanism".

Personally, I'd consider this as a feature quite similar to "Thumbs.db", another directory-level caching mechanism. Seems someone in Microsoft answered Raymonds questions before. And yes, I understand that thumbs.db is not part of NTFS. I'm just pointing out that an answer to the original question could have been "because there are more appropriate parts of the OS".

Derlin
17 Feb 2009 12:21 PM
#

I imagine that the average home user wouldn't care if this information was available to everyone and that hard links might not be accurately represented. However, I also imagine that certain businesses would very much want to restrict as much information as possible on files to users who own or have reason to access them. On a local desktop this would likely only be relevant to Documents and Settings, but on network shares could be just about anything.

Paul M. Parks
17 Feb 2009 1:16 PM
#

"I suppose the feature should be turn-off-able for those who DID mind showing this info."

I'm pretty sure Raymond talked about this recently:

http://blogs.msdn.com/oldnewthing/archive/2009/02/13/9416485.aspx

Gabe
17 Feb 2009 1:43 PM
#

Seeing as how an NTFS directory is just an index, it is quite likely that it does actually know how many files (entries) are in the directory. Anybody who is allowed to enumerate the files would have permission to get that number. Of course this is of no use to the person who wants the recursive number of files.

Aaargh!
17 Feb 2009 1:44 PM
#

"Considering the OWNER user is most likely to need the information (as he's likely the maintainer of the system)"

No. the most likely situation is that the owner(s) (shareholder(s) of a company) never use the system at all. And the most likely maintainer of the system is the IT department, not the one using it on a daily basis.

But this doesn't answer this blog entry's title: "Why doesn't the file system have a function that tells you the number of files in a directory?"

Even without storing this metadata it could still provide a convenience method for getting the directory size. Maybe not in the low-level system calls but in one of the system libraries. I image there are several interesting ways in which someone could screw up the task of recursively determining a directory's size (e.g. symlink-cycles) so

why not have a proven-correct implementation generally available ?

cjm
17 Feb 2009 1:57 PM
#

This is slightly off-topic, but the information disclosure thing reminds me of the way processes are handled. In XP, I find that whoever is logged in sees all processes in Task Manager, but if you have fast user switching then subsequent users don't get the full list. They only see their own processes.

Another problem is that if a program is searching the process list, access is not restricted to processes by the current user. So if I want to fire up explorer with a runas command, it will check to see if there is another copy running (which there will be, under MY own account) and quit. Firefox does the same thing. This just reminds me of the information disclosure thing that Raymond mentioned. I would have thought that a user shouldn't be able to determine what others are running, unless they have permission.

Anonymous
17 Feb 2009 2:24 PM
#

> Indeed the added cost of all the metadata writes is one of the reasons why Windows Vista no longer updates the Last Access time by default.

Updating the last access time is even worse than most examples, since it converts what should be nothing more than a simple read (which can be heavily cached) into a simple read plus a much more expensive metadata update. The difference can be dramatic if the data is on a disk which has been spun down to save power and the read could be satisfied from the cache.

On Linux, most distributions seem to be enabling "relatime" lately. The relatime mount option causes the last access time to be updated only if its current value is less than or equal to the modification time (basically, turning it into a "this file has been read since it was last written" flag). There is also "noatime" (which completely disables the updates), but a few programs need at least the "relatime" behaviour.

Aaron
17 Feb 2009 2:54 PM
#

I think you're looking at it from a Service-perspective, rather than a Customer-perspective.

I agree that NTFS probably cannot provide this information easily and consistently.

But customers still have a legitimate want for this data.

The Indexing-Service should be able to provide it.

Indexing already keeps track of EVERY SINGLE WORD IN EVERY SINGLE FILE! But

somehow, can't yet tell me how big my files are, or how many files are in a director?

Based on the volume and type of information that indexing tracks, I think it could also track information on files regarding numbers, and sizes, properly accounting for permissions and hard-links.

Ultimately, this information should be available to the Customer. I don't care which service provides it.

> [*If you want to store this information outside the file system, then more power to you. -Raymond*]

Leo Davidson
17 Feb 2009 3:03 PM
#

@cjm:

Explorer is unlikely to be checking the process list, and you can easily verify that you *can* run multiple copies of Explorer under different users (else Terminal Services and Fast User Switching would be a little broken).

It's probably looking for a named object in the Local\ (session) scope to avoid creating more than one instance of itself in the same session.

@Michiel:

I don't think it would make sense for directory size information to be stored similar to thumbs.db because validating that the cached size information was up-to-date would often take as long as re-calculating the information. You'd have to do a scan of the directory structure to see if any sub-directories had changed or been added since the cache file was written.

(You might be able to exclude some directories at the deepest part of the tree by noticing that their timestamps haven't changed since the cache was last updated, but that only works for folders you know to have no sub-folders. The modified timestamp doesn't change on a folder if its sub-folder contents are changed; only if it's own contents are changed. Even then I'm not sure if it's reliable, and there's definitely the problem of programs (re-)setting folder timestamps (which could also affect thumbnails, of course).)

FWIW there's a shell extension for Explorer that runs a SYSTEM service which keeps track of folder size changes. Some people seem to like it. Personally I think the idea is a bit flawed as it can never be completely accurate (plus keeping that info in memory seems a waste given how often one typically wants to see folder sizes in such a hurry) but I guess if you don't need that, and/or don't care about the cases where it'd be inaccurate, then it does work.

BCS
17 Feb 2009 3:52 PM
#

Is their any MS provided code for computing these things? I would expect that even if

the FS doesn't have it built in it would be nice to be able to get the info out of the box. Heck Explorer must have the code somewhere.

Ken Hagan
17 Feb 2009 4:02 PM
#

"I suppose the feature should be turn-off-able for those who DID mind showing this info."

APIs that may or may not work really suck. Any programmer who wants to use the information has to check whether it is enabled (and even if it is then the value might not be accurate) and so if the feature is used for any non-cosmetic purpose then they need to code an alternative algorithm, which they might as well then use on all platforms, unless it is really expensive, because maintaining one body of code is always easier than maintaining two.

If you want a purely-cosmetic-because-it-might-not-be-accurate number to appear in Explorer, use rand().

Peter
17 Feb 2009 4:04 PM
#

@Michiel:

Please no, not another file like that poxy thumbs.db that ends up everywhere for no good reason (yes, I know you can turn it off in Explorer; no, that doesn't stop them from getting onto my USB key from other people's machines).

Microsoft aren't the only ones to do this, nor the most egregious. But it's still a little pet peeve (and, IMO, a good case of "don't keep track of what you don't need").

As has been pointed out it just replaces one problem (how do I get the size of this directory?) with another (how do I verify that size.db is up-to-date?) which isn't even significantly easier.

laonianren
17 Feb 2009 4:26 PM
#

@Aaron: Indexing service can tell you how big files are. It stores a variety of metadata, including the size.

I think you can also find out how many files are in a directory with an appropriate query.

But since it doesn't precalculate totals, I suspect it would be quicker to get this information from the file system.

Alexandre Grigoriev
17 Feb 2009 4:41 PM
#

@cjm,

Task Manager shows the processes it can enumerate. If you have administrative privileges, it will show you all terminal sessions. If you don't have admin privs, it will only show you your session. If you're the first user logged in, it will also show you all system services (which run on session 0). In vista, no user runs on session zero, and you won't see all services.

Ray Trent
17 Feb 2009 4:48 PM
#

In addition to the security considerations, the other problem is that any such value would have to have too many caveats applied to it to be a useful number.

And storing useless information is even worse than storing information you don't need.

For example, consider the cases of network shares, which can be updated by other machines even when your machine is off (especially with NAS drives).

Even if you only wanted this mythical value to track "directories on disks on the local machine" (assuming you can really even figure that out, given the presence of local mappings and substs), how about directories that have links to files or subdirectories on network drives? Should their number only refer to "the subset of the stuff in this directory that 'I' happen to have been the one to have written 'locally'"?

Now, should their be a system call that accurately *calculates* this number? That's a useful question to ask.

dave
17 Feb 2009 5:28 PM
#

The question about 'ownership' is not entirely correct. There is some confusion here about people like Raymond who use 'owner' as 'the user identity that owns the file' and others who seem to think it is 'the real person who paid for the computer'.

Fortunately, we don't have to resolve that question.

The real question is 'who is allowed to see the metadata' such as file size. The ACL of a file gives precise control over this via the READ_ATTRIBUTES access.

Apart from that quibble, I'm on the side of people who say it's no business of the file system to maintain running totals of the space consumed by the subtree rooted at some arbitrary directory. How often is that needed, anyway? It is programmatically simple to count it when needed.

Aaargh!
17 Feb 2009 6:18 PM
#

Another difficulty with doing this: what happens to you directory size when you mount either another filesystem or a network drive somewhere in your directory tree ?

Jeff Tyrrill
17 Feb 2009 8:04 PM
#

@DWalker59:

"My computer doesn't HAVE any non-owner users. Ever. But no, in general, I wouldn't mind at all.

I suppose the feature should be turn-off-able for those who DID mind showing this info."

So Microsoft should punch holes in the low-level security system because you think that some owners wouldn't "mind" sharing this information, and it's the burden of every owner to track down all of these holes and re-enable the security unless they agree with these arbitrary proposed inconsistencies in the security system?

That reminds me of somebody complaining once in an article (sorry, I'll never be able to locate it, it was years ago) that the Music folder which is currently in My Documents ought to instead be a single shared folder on the machine, because they believed typically in a home everybody would want to share their music with each other. That's not a valid justification for destroying the per-user/per-computer separation which is an architectural principle of the operating system. If people in a household (or any other arrangement) would like to share music they are free to do so by creating a shared folder under the current system.

Pete
17 Feb 2009 9:09 PM
#

If you want an accurate count of just the number of files in a directory, (and it SEEMS like it would be easy to implement), you aren't thinking about multi-tasking. You'd have to have the OS block access to the directory while the calculations are done. We can't have files created or removed during the counting. Plus, the number your process got may not be accurate as after the count is done other processes can modify the directory. It is better to have a get-next-entry API for this reason.

Jolyon Smith
17 Feb 2009 9:27 PM
#

"APIs that may or may not work really suck"

Right, but how about an API that works all of the time but which may work one way or

another depending on environmental factors?

A lot of the "Yes but..." arguments against the file system maintaining this information simply do not apply to a home user or in many cases even to (the local files system) of an office computer used by a single individual.

If there was an API that yielded the information of interest, that API could take a look around it's environment and determine the appropriate mechanism for yielding the required result.

In a single-user, no security complications scenario, the file system maintained information could be used to quickly return the required result.

In a multi-user, security complicated scenario the figure would be calculated dynamically w.r.t context, just as ALL software currently has to do today.

Hard links etc certainly add further complication, but again, most people don't even know that such things exist, let alone use them, and again they could be allowed for within the API implementation, one way or another.

Pete
17 Feb 2009 9:44 PM
#

@Jolyon,

Even on local drives, in directories designated as belonging to the end user, having the OS keep track of per-folder disk use would seriously impact computer performance in a lot of common situations. In a scenario where you have N-deep directories, each single write by an application (of which there could be many per second), would require N-writes. A different process which shared any part of the path would have to acquire some sort of exclusive lock on the shared directories, wasting even more time and disk throughput. Heck, just moving a single file from one folder to another might make one have to worry about deadlocks! Personally I'd rather just keep the performance.

Kaenneth
17 Feb 2009 9:52 PM
#

My issue with Thumbs.db is that it is tagged as a system owned file (at least on XP)... every time I move/delete a media folder I get prompted if I want to move/delete the system file thumbs.db...

otherwise, I like having thumbnails, and it's good the metadata is stored with the data. A while back I ran a popular linux desktop, and when I switched back to Windows, used a extfs driver to copy my files back to a windows machine, and found many, many things I had deleted, and wanted very deleted, had thumbnails stored in a central .directory that wasn't cleaned when the origional files were deleted, including confidental faxes. (not a linux fault, I've seen the same thing with windows apps.)

Thumbnails, Sine/Cosine Tables, and other expensive to regenerate data is acceptable to retain; but retaining by default is a case of premature optimization, having to recalculate a few numbers in memory thousands of times over is much faster than swapping because

you ran out of ram.

Aaron Margosis
17 Feb 2009 10:22 PM
#

I didn't realize that about Vista and the last-accessed time stamp. Running as standard user on XP actually generates a lot of "access-denied" noise in a Process Monitor or LUA Buglight trace (the latter filters it out by default) because exe and dll files are opened for GenericRead + WriteAttributes so that last-access can be updated. Windows XP "degrades gracefully" by opening the file for just GenericRead and not updating last-access in that case.

Michiel
18 Feb 2009 3:40 AM
#

Glad to see I'm not the only one that dislikes the thumbs.db littering. I agree 100% with Raymond's remark to Aaron. This kind of information should be tracked by an optional service, and stored centrally.

Flushing the information to disk can be done lazily, as the information is fully recoverable after a crash. Of course, for efficiency reasons you'd want to keep dirty flags at a more granular level than entire disks. But you'd want to have some locality of reference to make recursive queries fast. So, you'd probably want to keep that data together. E.g. Program Files\ and My Documents would each have their own chunk of directory metadata info, with separate dirty flags.

Can this service be written outside Microsoft? Probably not, as you need to keep track of file changes during shutdown. That means that you'd need to keep write to disk after the NTFS susbsystem has shut down, but before the disk drivers themselves call it quits. And NTFS has to give you a bunch of fixed sectors that you can write to after it quit. I understand that there's similar functionality anyway for hibernating.

[*Um, use the change journal. That's what it's for: Tracking changes to the file system when you're not running. -Raymond*]

Mike Dimmick
18 Feb 2009 5:24 AM
#

@cjm, Leo Davidson: It's simply looking for a window with the appropriate window class, and sending it a message. If there's an IE or Firefox window already in your window station, it will be found and the new window will close.

If you don't want this to happen, turn off the 'Reuse windows for launching shortcuts' option (in IE).

Neil
18 Feb 2009 6:28 AM
#

Rather annoyingly, the last access date in an Explorer properties dialog is always "now", presumably because it just accessed the file in order to populate the Summary tab...

DWalker
18 Feb 2009 11:34 AM
#

@Ray Trent: "any such value would have to have too many caveats applied to it to be a useful number".

No, it would NOT have to have too many caveats to be useful! The information IS useful. If I know that it's not 100% correct if I have hardlinks, and I also know that I don't have any hardlinks (in my user directories), then I know that the number is useful. I don't mount other filesystems in my directory tree.

I am perfectly aware that what applies to me doesn't apply to everyone, but that doesn't make the information "not useful" to me.

As the creator of one of the free services said, in the original linked entry, he created something that was useful to him, and other people could use it if they wanted to. You can't unilaterally declare that the information is not useful.

And, when you discover that there are gigabytes of junk in a Temp folder, or gibgabytes of junk in the "desktop" or "documents" folder of a user whose logon name has been deleted, or gigabytes of files in a folder tree for a program that you uninstalled a long time ago, then you have gained some useful information.

Leo Davidson
18 Feb 2009 12:26 PM
#

@DWalker59:

How often do you check for wasted space? Unless you're doing it more than once an hour I don't see why you'd need an API that returned the information faster than it can be calculated already.

Some of the replies in this thread make it sound like it's *impossible* to find out how big directories are at the moment but that's not true at all. It's a trivial amount of code. One short, recursive function. Even if you check for links it's not complicated.

It's not _that_ slow, either. Unless you have something crazy -- like folders with 100,000 files in them, in which case you've probably got bigger problems and/or know about those special folders and can exclude them from measurements -- it's not going to take more than a few seconds to calculate the disk space used without needing any kind of cache.

That goes double for things like Temp, Desktop and Documents which are typically shallow folders which can be enumerated (recursively) in under a second each.

Also, if you are frequently re-scanning the same folders then it gets even quicker as the filesystem caches the data in memory. Open the Properties dialog for a folder with a lot below it, let it calculate, then close that an open it again. The second one will usually calculate almost instantly because it's just re-reading the filesystem structures from memory.

So given that it's easily calcualted now, in a reasonably quick way, and caching the data has all sorts of issues attached with it (whether or not you care about them, unaware people will run into them if a caveat-filled API were created), I don't see that it would be worth it at all for the filesystem to provide the data.

(Assuming the current type of filesystem, of course. If we one day move to a database filesystem then things might be very different.)

Robert Obryk
18 Feb 2009 1:10 PM
#

So what does the change journal do in presence of hardlinks and events associated with file itself and not its directory entry (for instance, file truncation)? Does it store event for every filename or for the one which was used when opening the file? If the former, then the list of file entries corresponding to a file must be stored with it on disk -- is there a public API to access it?

Yuhong Bao
18 Feb 2009 3:22 PM
#

Aaargh! was nitpicking the word "owner", perhaps admin would have been a better word, though often the admin is the owner, which reminds me of "The admin is an idiot" article from this blog.

Jolyon Smith
18 Feb 2009 5:01 PM
#

For the simple case where a directory size metric could be maintained (no security, single user etc), the "size of files in this directory" metric would be just that.... the size of files in THIS directory, not "this directory and all sub-directories".

A change to a file (or files) in a directory would affect the metric for THAT directory only.

The API for calculating the "size" of a directory would then still have to iterate over the sub-directory structure, but it would still be vastly more efficient than having to do that iteration over directories AND files in order to calculate the size. It would simply collect and sum the directory size metrics directly.

And since I suggested that any such API would internally select a strategy appropriate to the environment, then *if* performance were significantly impacted (I doubt that it

would if you maintained the metrics as described above), but *if* it was (or you felt/feared that it was), it would be trivial to have a "manual override" setting suppressing the maintenance of this metric and forcing the API to calculate the metric dynamically on request.

Dean Harding
18 Feb 2009 6:08 PM
#

@DWalker59: Nobody is saying that a program which keeps track of folder sizes isn't useful to *anybody*. But surely you can understand that it's not going to be useful to EVERYbody. In fact, it's only going to be useful to a very small percentage of people.

This blog post is about whether or not NTFS should keep track of recursive folder sizes for you. If such a feature is only going to be useful for a small percentage of people, and keeping track of those sizes has a performance penalty, then surely you must agree that this feature is better left to a third party program, rather than building it into NTFS where everybody would have to suffer the consequences?

required
18 Feb 2009 9:46 PM
#

Some of the reasons used to justify not storing the file size reminded me of the article Why did the Add or Remove Programs control panel try to guess all that information?

Not to say there's any immediate relevance, it just, reminded me.

Aaargh!
19 Feb 2009 8:20 AM
#

"Aaargh! was nitpicking the word 'owner', perhaps admin would have been a better word"

My point was that the admin/owner usually isn't the one using the computer on a daily basis. At least not in a corporate setting.

Leo Davidson
19 Feb 2009 12:09 PM
#

@Jolyon Smith:

"The API for calculating the "size" of a directory would then still have to iterate over the sub-directory structure, but it would still be vastly more efficient than having to do that iteration over directories AND files in order to calculate the size."

What makes you say that? How do you enumerate the sub-directories without also enumerating the files?

DWalker
19 Feb 2009 12:32 PM
#

@Dean Harding: You're right, although I always wished it was something that Windows would keep track of for me. Now I realize that perhaps Windows shouldn't; instead, users who want this should have to go through the process of finding a third-party tool, downloading it, trusting it, and installing it!

:-)

余啊雷
20 Feb 2009 5:48 AM
#

There are any number of bits of information you might want to query from the file system, such as the

Maybe insane, but who, really I?
20 Feb 2009 12:35 PM
#

I DON'T WANT filesystem to keep this info for me. I AM NOT TOO LAZY to call System.IO.Directory.GetFiles() and sum their sizes easily. I'm not too lazy to do it recursively for each GetFolder(). PowerShell can do it for me. And if I need it often, I'll just create a cmdlet in PowerShell. Same if I need number of files/subfolders.

Why all you commenters put so much energy into inventing totally useless crappy feature?? Wondering about all problems, I/O, performance, security, mountpoints, hardlinks, USN change journal and so on...

HOW OFTEN do you need this info?? And now again: how often do you REALLY need this info??

You aren't skilled to write yourself a script that would do that for you, but you feel skilled to resolve all the problems that arise from this stupid idea?? Then wow.

I DON'T THINK that people at Microsoft should spend ANY SECOND MORE wondering whether to implement this. DO NOT IMPLEMENT THIS.

About last access time, I think turning it off by default is A VERY GOOD IDEA. I'm turning it off at ANY MACHINE I administer since NT 4.0. I DIDN'T NEED IT A SINGLE TIME!! This is A DIFFERENT ISSUE than the above, because these times cannot be inferred, and I can imagine somebody may need it, BUT THEN HE/SHE SHOULD CONSCIOUSLY TURN IT ON!!

Friday
20 Feb 2009 2:37 PM
#

Just like with the start menu, I expect in the next reincarnation of Windows the current behaviour to be thrown out and only this hypothetical size information to be maintained.

Quote Aaron Margosis in previous thread (which is now closed for comments):

Keeping AND MAINTAINING older code involves a lot of risk, and sometimes that risk is not worth the marginal benefit.

Aaron Margosis
20 Feb 2009 11:56 PM
#

I would like to clarify and extend my previous comments:

Keeping and maintaining REDUNDANT AND/OR OBSOLETE older code involves a lot of risk, and sometimes that risk is not worth the marginal benefit.

Friday
21 Feb 2009 7:23 PM
#

Aaron, nothing personal, I was just upset with the probable removal of Classic start menu in Windows 7. (And I still can't calm down).

Igor Levicki
23 Feb 2009 2:08 AM
#

Why the filesystem can't just keep subtotals of each folder (not including subfolders) as a metadata updating them as the file sizes change and calculate grand-total of the recursive tree only when user clicks on properties to check it out? It would have a lot less work to do that way because it would just have to sum the folder sub-totals and not check each individual file.

As for the hardlinked files, the only logical way is to add their sizes only to the folder in which they physically reside, because that is where they are taking up the space. Hardlinks are most often made to make files more accessible in the hierarchy so usually the intention is not to copy them with a folder. Perhaps their size could be shown in a second sub-total on the property sheet, for example:

Folder size : 3,541,365 bytes

Folder size (with hardlinks) : 4,158,678 bytes

As for mounted filesystems, be it on a network or on another drive -- they do not take up the space on the drive on which the mountpoint resides. Therefore they should not be summed up with the rest, but you could also show two totals:

Folder size : 3.2 GB

Folder size (with mountpoints) : 41.5 GB

You folks seem to have rather poor imagination.

Igor Levicki
23 Feb 2009 2:11 AM
#

"I DON'T WANT filesystem to keep this info for me. I AM NOT TOO LAZY to call System.IO.Directory.GetFiles() and sum their sizes easily."

Yes, and have users pull their hair out waiting for the result.

Moreover, at least 80% of users don't know how to call System.IO.Directory.GetFiles().

Who asked you for an opinion anyway?

余啊雷
1 Mar 2009 2:28 AM
#

PingBack from http://blog.openquality.ru/software-quality-news-0209/