How does Explorer calculate “Size on disk”?

devblogs.microsoft.com/oldnewthing/20160427-00

April 27, 2016

Raymond Chen

When you ask to see the size of a file, you get two values. One is the nominal file size (the value that shows up in a directory listing). The other is something called “Size on disk”. What is “Size on disk”?

The algorithm for “Size on disk” is as follows:

- If the file is sparse, then report the number of non-sparse bytes.
- If the file is compressed, then report the compressed size. The compressed size may be less than a full sector.
- If the file is neither sparse nor compressed, then report the nominal file size, rounded up to the nearest cluster.

Each of these values sort of makes sense in its own naïve way.

If a file is not compressed, then it occupies some integral number of clusters, so charge it for the clusters that it uses. This is accurate for FAT file systems, but it is naïve for NTFS, which has multiple stages of file growth. (Thre’s even a stage for very small files, where the file contents aren’t stored in a dedicated cluster at all.)

If a file is sparse, then Explorer reports the number of bytes of the file that are taking up space on disk. The spaces filled with virtual zeroes are not reported, since they aren’t occupying any disk space. They only take up bookkeeping.

Furthermore, the “Size on disk” is naïve because it doesn’t take into account any metadata for the file. The space on disk used to store the file name, last-modified time, the file size, the security information, and where on the disk the file contents can be found. And then there are volume journal entries, volume snapshots, and other things which the file contributes to by its mere existence. None of those are captured in the “Size on disk”.

The upshot is that the *Size on disk* value reported by Explorer tries to say something that makes sense, based on context.

Bonus chatter: Starting in Windows 8.1, the *Size on disk* calculation includes the sizes of alternate data streams and sort-of-kind-of tries to guess which streams could be stored in the MFT and not count them toward the size on disk. (Even though they really are on disk. I mean, if they're not on disk, then where are they?)

Raymond Chen

Follow