Why is the FAT driver called FASTFAT? Why would anybody ever write SLOWFAT?

devblogs.microsoft.com/oldnewthing/20141030-00

October 30, 2014

Raymond Chen

Anon is interested in why the FAT driver is called FASTFAT.SYS. "Was there an earlier slower FAT driver? What could you possibly get so wrong with a FAT implementation that it needed to be chucked out?" The old FAT driver probably had a boring name like, um, FAT.SYS. At some point, somebody decided to write a newer, faster one, so they called it FASTFAT. And the name stuck. As for what you could possibly get so wrong with a FAT implementation that it needed to be improved: Remember that circumstances change over time. A design that works well under one set of conditions may start to buckle when placed under alternate conditions. It's not that the old implementation was wrong; it's just that conditions have changed, and the new implementation is better for the new conditions. For example, back in the old days, there were three versions of FAT: FAT8, FAT12, and FAT16. For such small disks, simple algorithms work just fine. In fact, they're preferable because a simple algorithm is easy to get right and is easier to debug. It also typically takes up a lot less space, and memory was at a premium in the old days. An $O(n)$ algorithm is not a big deal if n never gets very large and the constants are small. Since FAT16 capped out at 65535 clusters per drive, there was a built-in limit on how big n could get. If a typical directory has only a few dozen files in it, then a linear scan is just fine. It's natural to choose algorithms that map directly to the on-disk data structures. (Remember, data structures determine algorithms.) FAT directories are just unsorted arrays of file names, so a simple directory searching function would just read through the directory one entry at a time until it found the matching file. Finding a free cluster is just a memory scan looking for a 0 in the allocation table. Memory management was simple: Don't try. Let the disk cache do it. These simple algorithms worked fine until FAT32 showed up and bumped n sky high. But fortunately, by the time that happened, computers were also faster and had more memory available, so you had more room to be ambitious. The big gains in FASTFAT came from algorithmic changes. For example, the on-disk data structures are transformed into more efficient in-memory data structures and cached. The first time you look in a directory, you need to do a linear search to collect all the file names, but if you cache them in a faster data structure (say, a hash table), subsequent accesses to the directory become much faster. And since computers now have more memory available, you can afford to keep a cache of directory entries around, as

opposed to the old days where memory was tighter and large caches were a big no-no. (I wonder if any non-Microsoft FAT drivers do this sort of optimization, or whether they just do the obvious thing and use the disk data structures as memory data structures.)

The original FAT driver was very good at solving the problem it was given, while staying within the limitations it was forced to operate under. It's just that over time, the problem changed, and the old solutions didn't hold up well any more. I guess it's a matter of interpretation whether this means that the old driver was "so wrong." If your child outgrows his toddler bed, does that mean the toddler bed was a horrible mistake?

Raymond Chen

Follow