If NTFS is a robust journaling file system, why do you have to be careful when using it with a USB thumb drive?

devblogs.microsoft.com/oldnewthing/20130101-00

January 1, 2013

Raymond Chen

Some time ago, I noted that in order to format a USB drive as NTFS, you have to promise to go through the removal dialog.

But wait, NTFS is a journaling file system. The whole point of a journaling file system is that it is robust to these sorts of catastrophic failures. So how can surprise removal of an NTFS-formatted USB drive result in corruption?

Well, no it doesn't result in corruption, at least from NTFS's point of view. The file system data structures remain intact (or at least can be repaired from the change journal) regardless of when you yank the drive out of the computer. So from the file system's point of view, the answer is "Go ahead, yank the drive any time you want!"

This is a case of looking at the world through filesystem-colored glasses.

Sure, the file system data structures are intact, *but what about the user's data*? The file system's autopilot system was careful to land the plane, but yanking the drive killed the passengers.

Consider this from the user's point of view: The user copies a large file to the USB thumb drive. Chug chug chug. Eventually, the file copy dialog reports 100% success. As soon as that happens, the user yanks the USB thumb drive out of the computer.

The user goes home and plugs in the USB thumb drive, and finds that the file is corrupted.

"Wait, you told me the file was copied!"

Here's what happened:

- The file copy dialog creates the destination file and sets the size to the final size. (This allows NTFS to allocate contiguous clusters to the file.)
- The file copy dialog writes a bunch of data to the file, and then closes the handle.
- The file system writes the data into the disk cache and returns success.

- The file copy dialog says, “All done!”
- The user yanks the USB thumb drive out of the computer.
- At some point, the disk cache tries to flush the data to the USB thumb drive, but discovers that the drive is gone! Oops, all the dirty data sitting in the disk cache never made it to the drive.

Now you insert the USB drive into another computer. Since NTFS is a journaling file system, it can auto-repair the internal data structures that are used to keep track of files, so the drive itself remains logically consistent. The file is correctly set to the final size, and its directory entry is properly linked in. But the data you wrote to the file? It never made it. The journal didn’t have a copy of the data you wrote in step 2. It only got as far as the metadata updates from step 1.

That’s why the default for USB thumb drives is to optimize for Quick Removal. Because people expect to be able to yank USB thumb drives out of the computer as soon as the computer says that it’s done.

If you want to format a USB thumb drive as NTFS, you have to specify that you are Optimizing for Performance and that you promise to warn the file system before yanking the drive, so that it can flush out all the data sitting in the disk cache.

Even though NTFS is robust and can recover from the surprise removal, that robustness does not extend to the internal consistency of the data you lost. From NTFS’s point of view, that’s just a passenger.

Update: It seems that people missed the first sentence of this article. Write-behind caching is disabled by default on removable drives. You get into this mess only if you override the default. And on the dialog box that lets you override the default, there is a warning message that says that when you enable write-behind caching, you must use the *Safely Remove Hardware* icon instead of just yanking the drive. In other words, this problem occurs because you explicitly changed a setting from the safe setting to the dangerous one, and you ignored the warning that came with the dangerous setting, and now you’re complaining that the setting is dangerous.

Raymond Chen

Follow