The Old New Thing

Why does copying a file to my USB thumb drive say that the parameter is incorrect?

11 Oct 2011 7:00 AM | 40

Consider the following sequence of operations, assuming that F: is a USB thumb drive with plenty of disk space.

```
C:\Users\Bob\Downloads> copy readme.txt F:\
        1 file(s) copied.
C:\Users\Bob\Downloads> copy Update.iso F:\
The parameter is incorrect.
```

Why is the second file copy failing?

The hint is the file extension: *.iso, which suggests that this is a CD or DVD image, and DVD images have the feature that they tend to be really big.

Like more than 4GB big.

USB thumb drives tend to be formatted with the FAT32 file system rather than with NTFS. And FAT32 has a maximum file size of 4GB minus one byte.

The user confirmed that the `Update.iso` file was larger than 4GB and that the USB thumb drive was formatted as FAT32.

Mind you, the error message doesn't help at all in identifying that this is what's going on. I don't know where it's coming from, but my guess is that somewhere inside the `copy` command, it tries to create the destination file and set its file size. Since the file size is out of range for FAT32, the call fails with the error `ERROR_INVALID_PARAMETER`, and that's what ends up bubbling out to the user.

But at least now you know what the confusing error message is trying to tell you.

Blog - Comment List MSDN TechNet

Comments

alegr1
11 Oct 2011 7:16 AM
#

FAT(32) needs to go away in favor of exFAT. Unfortunately, there are reasons for the manufacturers to use only FAT32 as an initial format.

ANother "helpful" error message is when a storage port returns SRB_STATUS_TIMEOUT. It gets translated to STATUS_IO_TIMEOUT, which gets translated to ERROR_SEM_TIMEOUT, and gives: "The semaphore timeout period has expired.". Lolwut?

JM

11 Oct 2011 7:49 AM
#

I format my large USB sticks as NTFS for this reason. The primary reason I *have* sticks that big is to store large files, not to store lots of small files.

Of course, NTFS on a removable drive is fraught with peril, as detailed on this very site (blogs.msdn.com/.../108205.aspx). But if you play by the rules, it works fine.

I didn't know exFAT existed, so that's good information. Linux support is still sketchy, it seems, which is a reason for me to favor NTFS for now.

Cesar
11 Oct 2011 8:06 AM
#

Another option is UDF: superuser.com/.../using-udf-on-a-usb-flash-drive

Unless you are trying to write to it from Windows XP (which should cease to be a problem in a few years), it works quite well in my experience. (Windows XP can read, but not write.)

Joshua
11 Oct 2011 8:17 AM
#

@alegrl: Problem 1: No Windows XP Support. Portable media must support XP for years to come. Problem 2: Patents. Removable media filesystems need to be practically not patent protected. The patents on VFAT will probably fail due to latches, but the patents on exFAT look good.

Tanveer Badar
11 Oct 2011 8:32 AM
#

Solution: convert /?

Been doing that for ages with USB thumbies.

alegr1
11 Oct 2011 9:20 AM
#

@Joshua:

www.microsoft.com/.../details.aspx

Jeff
11 Oct 2011 9:25 AM
#

Instead of using an unportable filesystem, you could always just split the file into multiple parts. (Unfortunately, Windows doesn't come with anything like "dd" as far as I know, but archiver programs can create multi-file archives, e.g. 7-zip's -v option)

SimonRev
11 Oct 2011 9:37 AM
#

Ironically this bit me while trying to install the Windows 8 developer preview. As I didn't have a blank 9 GB dvd, I tried to put the install onto a thumb drive. Unfortunately one of the files was in the ~5GB range and it failed (actually the software I was using didn't even report the failure -- I discovered it during the install process).

Eventually I found another bit of software that presumably formatted my USB stick with exFAT before copying the files.

voo
11 Oct 2011 9:47 AM
#

Yeah using exFat is a great idea.. well except if you have to use other users linux workstations (yes I know there's some support, but it doesn't seem that stable and reliable), macs or Win XP installs that didn't go through the length of installing an optional update (ie basically anyone's)

So it's basically unportable and NTFS has its own share of problems which makes me a bit uneasy with it, so yes FAT32 is still the best solution in my opinion. Well "best" - let's say least worst.

Anonymous
11 Oct 2011 9:58 AM
#

I have also seen the error code ERROR_DISK_FULL come up in this context.

alegr1
11 Oct 2011 9:58 AM
#

Just to clear misunderstanding:

"I found another bit of software that presumably formatted my USB stick with exFAT"

Vista+ format/GUI format can do that.

Gabe
11 Oct 2011 1:48 PM
#

I like to be able to take a USB stick and plug it into my car's radio to play MP3s, plug it into a kiosk at the photo lab to print JPEGs from it, plug it into my neighbor's Mac to transfer files with it, plug it into a computer at the hotel's business center to print documents, use it to copy photos onto my mother's digital picture frame, use it to show PPTs on video projectors, and also use it to copy utilities onto whatever computer I'm trying to fix.

It will be a decade or more before I can expect all of those systems to support anything other than FAT32.

GreenCat
11 Oct 2011 2:08 PM
#

Wasn't it better to establish an intelligible error code newly?

Crescens2k
11 Oct 2011 2:09 PM
#

I have seen Invalid Parameter come up in some other interesting contexts too.

One of them was that I was creating a catalog file for a Windows setup image, SIM was extracting files to my temporary directory but I accidentally let it run out of space. The files that it extracted ended up pretty much undeleteable because of this. For some reason, every time I tried doing something with those files it returned this error. If I tried to copy somewhere else, it produced this error, if I tried to delete then it produced this error. This was even with an administrator account and the exact same thing occured under explorer and cmd.

In the end, I had to boot into Windows PE to delete these files.

Pete
11 Oct 2011 2:11 PM
#

I keep software ISO's on a memory stick so I can quickly install my common programs. I have hit this problem a few times, tried NTFS but you have to be so careful with it. exFAT is a vast improvement just need better support in other OS's.

Leo Davidson
11 Oct 2011 3:07 PM
#

One warning about exFAT: Don't build Visual Studio 2010 projects on exFAT drives. That combination can result in randomly corrupted binaries.

The bug is in VS2010, not in the exFAT filesystem, so it's not a reason to distrust exFAT (I use it myself for large removable drives where I don't care about compatibility with other OS). It's just something to be aware of and a combination to avoid. It certainly caused me some headaches.

Nick
11 Oct 2011 3:29 PM
#

I've always been sad that Windows can put split files back together (with copy) but cannot split them.

Anonymous Coward
11 Oct 2011 4:40 PM
#

Leo, why is that? Why would VS2010 care about what file system is used? Or how, even?

ender
12 Oct 2011 3:01 AM
#

Using removable drives with alternate filesystems would've been so much easier if Windows didn't treat drives with removable flag set as special and didn't prevent you from partitioning them. Unfortunately it does, and finding USB sticks that don't have the removable flag set is nearly impossible (what I don't understand is, why Windows treats the removable flag as special - it has nothing to do with the drive actually being removable, as I learned when I tried installing XP on a CF card in IDE->CF adapter).

Leo Davidson
12 Oct 2011 4:13 AM
#

@Anonymous Coward:

I don't know how VS2010 manages to have such a bug but it has been confirmed. There's more info here:

www.pretentiousname.com/.../index.html

connect.microsoft.com/.../vs2010-produces-corrupt-binaries-for-projects-on-exfat-

partitions

Gabe
12 Oct 2011 7:34 AM
#

Leo: From the Connect page, it seems the problem is actually related to incremental linking. It only seems to be exFAT-related because it's easy to reproduce on exFAT (and maybe FAT), but very difficult to repro on NTFS.

alegr1
12 Oct 2011 7:48 AM
#

@Gabe:

Must be caused by the last modification time granularity.

JamesJohnston
12 Oct 2011 10:08 AM
#

Pardon my ignorance, but why is NTFS so finicky about being on a removable drive? Shouldn't it be *more* robust and not less robust on a removable drive? My computer's internal NTFS-partitioned hard drive doesn't get scrambled when the computer crashes / loses power. Why would that happen to my NTFS USB stick on surprise removal?! (I thought NTFS was designed to have more robust data structures than FAT, etc. etc. I can't remember the last time I had severe file system corruption with NTFS on my Windows drive, if ever.)

Tanveer Badar
12 Oct 2011 11:29 AM
#

My guess. Screw with a transactional system's log and it screws you back.

JamesJohnston
12 Oct 2011 1:52 PM
#

@Tanveer: right - NTFS is a journaling file system. How does a surprise USB removal screw with its log? "It is a critical functionality of NTFS for ensuring that its internal complex data structures, and indices will remain consistent in case of system crashes" (--Wikipedia). Which is true with any similar file system. Or so I thought.

I'm genuinely curious how this can be - surprise removals when the USB drive is FAT should be causing corruption - not NTFS - right??

Something else is going on and I'd love to know what.

> [*That "something else" is usability. Users are accustomed to yanking USB drives as soon as the copy dialog disappears. If they yank it before the journal flushes, then the drive is still intact, but the last few bytes of the file did not get copied. - Raymond*]

Evan
12 Oct 2011 2:02 PM
#

@James:

(I don't have direct knowledge about this, but I think this is a good guess. I'll talk as if it's true.) It's a combination of factors. If you follow the link JM posted, you'll see there are options for "optimize for speed" vs "optimize for quick removal". The former is what you're supposed to use with NTFS, but it will enable write caching. That write caching is why you can lose data, because it might not actually be flushed out to disk.

File system corruption itself is a different beast, and NTFS will be better at that -- and you won't get it with a USB drive any more likely than with a hard drive. However, this refers to file system metadata: what blocks are associated with which files, link counts of files, etc. This is what you get from the NTFS journal -- not protection that the data you are writing is on disk, but that the file system structures remain consistent.

Long story short: NTFS should make you more robust from a file system point of view. However, the fact that apparently you may need to turn on write caching to use it means that you can lose data.

NB
13 Oct 2011 4:57 AM
#

Doesn't everyone use "Safe Removal" from System Notification Area when removing an USB thumb drive?

That should ensure that you never lose data even if the file system is NTFS.

JamesJohnston
13 Oct 2011 6:53 AM
#

"options for "optimize for speed" vs "optimize for quick removal". The former is what you're supposed to use with NTFS"

Right, it seems pretty obvious the latter option disables write caching. If the former is what you have to use with NTFS then the reasons for requiring safe removal should be obvious. It's good to know that at least the total file system doesn't get

corrupted/trashed, according to Raymond. It's reasonable to expect an individual file recently written to have issues. Not reasonable for the file system to be trashed which was my initial impression upon reading about this requirement.

I don't really see why the "quick removal" option *should* be incompatible with NTFS. Maybe the NTFS driver just does not support drives that don't have write caching and the business case wasn't made for supporting it? (I can't see a reason why the NTFS driver would care about how the underlying drive caching works, but then again I've never written a file system driver). And then the question: what if I change the setting for speed, format a USB stick as NTFS, and then move the drive to another computer that hasn't been configured yet. Does the drive refuse to mount until I change the setting? Does the setting change automatically? Or does NTFS run anyway in a safer configuration for quick removal? (Somebody with more time than me can try this!)

It seems to me that if NTFS supported "optimize for quick removal" then it would be an even safer choice than FAT. I could use it with the same quick removal style as a FAT32 drive. And if I yank the drive in the middle of writing all kinds of stuff to the drive, at least the file system should stay consistent and not make a mess of itself like FAT file systems might.

alegr1
13 Oct 2011 9:19 AM
#

I think NTFS FSD doesn't check if the volume is removable, for different file cache policies. FASTFAT driver does. For example, lazy write may be less lazy on a removable FAT volume. If FSD fails to consider removability, it may become very annoying, such as XP/Win2003 behavior with the USB drives. God forbid you unplugged them without invoking "safe removal". They will tell you about delayed write error on a metafile.

David Walker
13 Oct 2011 11:50 AM
#

That VS2010 bug is REALLY weird. Especially since Microsoft has confirmed it, and it IS exFAT-related. As several people pointed out, Visual Studio should not care about the file type of the underlying storage.

But as Leo points out on his page, mt.exe had a "FAT32 workaround" switch at one point. I'll bet there was some code in Visual Studio or mt.exe that did some things that Raymond has said on this very blog, that you should not do. :-)

640k
13 Oct 2011 3:42 PM
#

ntfs has been a transactional file system for many years. Not a single byte should be corrupt or missing, it should be foolproof.

Wby does the copy dialog disappear (or indicate operation finished) before a copy

operation is flushed on a removable file system?

> [*It finished writing the last byte. As far as the file copy is concerned, it's done. (Forcing a flush is in general a bad idea.) -Raymond*]

Gabe
13 Oct 2011 4:06 PM
#

640k: Transactions in NTFS are optional; they have to be initiated by the application. Even so, transactions ensure that data is not corrupt, not missing. In other words, the data will all be there (committed) or all missing (rolled back).

alegr1
13 Oct 2011 4:33 PM
#

"It finished writing the last byte. As far as the file copy is concerned, it's done. (Forcing a flush is in general a bad idea.) -Raymond"

In Vista+,if you copy a large file to a USB drive, and want to cancel operation, you may have to wait for long time. Because it's in overlapped mode, not cached, and has enormous amount of pending write IOs which are not cancelable. It doesn't need flush. Don't remember if Win7SP1 has it fixed.

But in older OSes, when the lazy write was used, it was making total sense to limit amount of dirty data for 1 seconds of writes worth, or so.

Someone
14 Oct 2011 1:48 AM
#

@Gabe: I'm not sure if 640k is talking about the fact that NTFS is a journaling file system since it was invented (which only protects metadata), or about the transactional support added with Vista that allows commit/rollback of file operations like a database. (I would not consider something introduced in Vista as being existent "for many years".)

I'm also not sure, if committing a transactional file operations really means that all data related to that operations (metadata and filedata) are flushed to disks during the commit. I see this file transactions only as a form of isolation: Other processes will see the transacted changes all at once, or never, but no intermediate changes.

Anonymous
14 Oct 2011 2:59 PM
#

Can't send mail to user: not a typewriter

Evan
15 Oct 2011 4:21 AM
#

@Someone: If NTFS transactions *don't* flush to disk (or at least have that as an option), I would view that as a major problem. Maybe I'm too much in a database mindset, but to me, "transaction" (at least in the context of disk operations) means ACID. If you lack a flush you don't even really get isolation, because if it crashes in the middle than a process that starts after the restart isn't isolated from the intermediate changes; for this among other reasons, IMO transactions without ACID buy very little. (Exception: the OS says "transaction committed!" but it is not flushed to disk. A crash happens. The state of the system after boot is in the pre-transaction state. This isn't ideal, but it's pretty good as long as if you want durability, you can stick a flush inside your txn.)

Fortunately, msdn.microsoft.com/.../cc163388.aspx indicates that it's ACID: "When I say transacted file operations, I am talking about transactions with fully ACID semantics".

640k
15 Oct 2011 7:06 AM
#

Forcing a flush from a GUI standpoint on *removable file system* is NOT a bad idea.

foo.bar
15 Oct 2011 2:40 PM
#

A bug the Windows team never bothered to fix

Klimax
15 Oct 2011 11:43 PM
#

@foo.bar:

What bug? Error in error message? (passing internal error)

Someone
17 Oct 2011 2:34 AM
#

@Evan: I found a better explanation here: msdn.microsoft.com/.../ee240893%28v=VS.85%29.aspx. "When a commit occurs, TxF first flushes all undo information, then flushes the actual file changes, and then writes

and flushes a commit record." etc etc

So yes, TxF not only provides isolation of ongoing modifications but also ensures the persistence of commits.

"Forcing a flush from a GUI standpoint on *removable file system* is NOT a bad idea." I also think so because this would perform better than doing all operations with a disabled write cache.