What does the /V (verify) flag to XCOPY mean, and how did it get that way?

devblogs.microsoft.com/oldnewthing/20160121-00

January 21, 2016

Raymond Chen

The `XCOPY` command has a flag called `/V`, which stands for *verify*. Sort of.

```
/V          Verifies the size of each new file.
```

First, the easy question: What does this flag do?

Answer: If this flag is set, then after a file is copied, the `XCOPY` program will verify that the source and destination files are the same size.

As you might surmise, this doesn't really get you much. On top of that, disk caching means that the file size it reads most likely did not come from the hard drive. It came from the disk cache. So it's essentially verifying that RAM is not corrupted, which is really not all that interesting.

But wait, why does this flag even exist, seeing as it's so lame as to be useless?

The `XCOPY` command got the `/V` option from the `COPY` command. You can even find remnants of that command today:

```
C:\> VERIFY /?
Tells cmd.exe whether to verify that your files are written correctly to a
disk.

VERIFY [ON | OFF]

Type VERIFY without a parameter to display the current VERIFY setting.
C:\> VERIFY
VERIFY is off.
```

What does it mean to "verify that your files are written correctly to a disk"?

What it means to `CMD.EXE` is that after copying each file, it goes back and rereads both the source and destination and performs a byte-for-byte comparison of the files. If any bytes differ, it reports an error.

And in the presence of disk caches, this comparison is largely useless, since both the source and destination are most likely still in the disk cache, so all this is doing is comparing two RAM buffers against each other, which has nothing to do with whether the file got written successfully to disk.

Okay, so why does `COPY` have this weird verify behavior?

All this rigamarole over "verifying" comes from MS-DOS. Normally, when MS-DOS wrote to the disk, it issued command 8 (write) to the device driver. But if you did a `VERIFY ON`, then MS-DOS would issue command 9 (write with verify) instead. Device drivers were expected to handle the "write with verify" command by writing the data to the disk, then reading the data back and comparing it to what should have been written.

At the end of the day, it was up to the device driver to do the verification. All that `VERIFY ON` did was set a flag that eventually made its way down to the device driver, who was expected to do something.

Okay, now let's run the time machine forward again. MS-DOS had this `VERIFY ON` thing. But in the new I/O model, there is no "write and verify" command. There's just "write". So `CMD.EXE` fakes it by doing its own verification: After copying a file, it reads it back. And `XCOPY.EXE` fakes it by simply checking whether the file sizes match. Neither of these fake verifications really accomplish much because of disk caching. But the options are there for backward compatibility.

Raymond Chen

Follow