The case of the asynchronous copy and delete

devblogs.microsoft.com/oldnewthing/20120907-00

September 7, 2012

Raymond Chen

A customer reported some strange behavior in the `CopyFile` and `DeleteFile` functions. They were able to reduce the problem to a simple test program, which went like this (pseudocode):

```
// assume "a" is a large file, say, 1MB.
while (true)
{
  // Try twice to copy the file
  if (!CopyFile("a", "b", FALSE)) {
    Sleep(1000);
    if (!CopyFile("a", "b", FALSE)) {
      fatalerror
    }
  }
  // Try twice to delete the file
  if (!DeleteFile("b")) {
    Sleep(1000);
    if (!DeleteFile("b")) {
      fatalerror
    }
  }
}
```

When they ran the program, they found that sometimes the copy failed on the first try with error 5 (`ERROR_ACCESS_DENIED`) but if they waited a second and tried again, it succeeded. Similarly, sometimes the delete failed on the first try, but succeeded on the second try if you waited a bit.

What's going on here? It looks like the `CopyFile` is returning before the file copy is complete, causing the `DeleteFile` to fail because the copy is still in progress. Conversely, it looks like the `DeleteFile` returns before the file is deleted, causing the `CopyFile` to fail because the destination exists.

The operations `CopyFile` and `DeleteFile` are synchronous. However, the NT model for file deletion is that a file is deleted when the last open handle is closed.[1] If `DeleteFile` returns and the file still exists, then it means that somebody else still has an open handle to the file.

So who has the open handle? The file was freshly created, so there can't be any pre-existing handles to the file, and we never open it between the copy and the delete.

My psychic powers said, "The offending component is your anti-virus software."

I can think of two types of software that goes around snooping on recently-created files. One of them is an indexing tool, but those tend not to be very aggressive about accessing files the moment they are created. They tend to wait until the computer is idle to do their work. Anti-virus software, however, runs in real-time mode, where they check every file as it is created. And that's more likely to be the software that snuck in and opened the file after the copy completes so it can perform a scan on it, and that open is the extra handle that is preventing the deletion from completing.

But wait, aren't anti-virus software supposed to be using oplocks so that they can close their handle and get out of the way if somebody wants to delete the file?

Well, um, yes, but "what they should do" and "what they actually do" are often not the same.

We never did hear back from the customer whether the guess was correct, which could mean one of various things:

1. They confirmed the diagnosis and didn't feel the need to reply.
2. They determined that the diagnosis was incorrect but didn't bother coming back for more help, because "those Windows guys don't know what they're talking about."
3. They didn't test the theory at all, so had nothing to report.

We may never know what the answer is.

Note

[1]Every so often, the NT file system folks dream of changing the deletion model to be more Unix-like, but then they wonder if that would end up breaking more things than it fixes.

Raymond Chen

Follow