Why are hidden files with a leading tilde treated as super-hidden?

devblogs.microsoft.com/oldnewthing/20170526-00

May 26, 2017

Raymond Chen

Open a command prompt and perform the following operations:

```
C:> cd /d %USERPROFILE%\Desktop
C:\Users\Bob\Desktop> echo 12345 > ~test.txt
C:\Users\Bob\Desktop> attrib +h ~test.txt
```

This creates a hidden file called `~test.txt` on the desktop. Configure Explorer to show hidden files. Observe that the `~test.txt` file does not appear.

But wait, there's more, if you're running Windows 7 (but not Windows 8 or higher): Configure Explorer to show both hidden files and protected system files. The `~test.txt` file will now appear, and it will be dimmed because it is hidden. Use `Ctrl`+`C` and `Ctrl`+`V` to create a copy of the file. Observe that the copy has both the hidden and system attributes, even though the original did not have the system attribute.

A customer discovered this behavior and wanted to know whether it was a bug or a feature (or a buggy feature).

There are multiple things going on here, so let's take them separately.

First, why doesn't `~test.txt` appear on the desktop even though Explorer is configured to show hidden files?

This behavior dates back to Windows Vista. If there is a hidden file whose name begins with a tilde, then Explorer treats it as if the system and hidden attributes are both set, causing the file to be treated as super-hidden. That's why you have to disable "Hide protected operating system files" in order to see them.

Why does this rule exist?

In practice, hidden files that begin with a tilde are temporary files, usually to represent auto-saved contents, or as part of a write-rename-delete save operation. These files are not intended to be user-manipulated, so Explorer treats them as super-hidden so that the user

won't be tempted to rename or delete them and mess up the operation of the program that created them.

Second, why does copying these artificially-super-hidden files cause the copy to become super-hidden for real?

This is a case where Explorer faked itself out.

The code that creates item IDs for files reads the file attributes and records them for future reference. It is this code that checks for the leading tilde and if found internally sets the `FILE_ATTRIBUTE_SYSTEM` flag on the item it created. This is what causes hidden files beginning with a tilde to be treated as super-hidden.

The problem is that this code ends up doing too good a job of fooling the rest of the shell. There is no flag anywhere that says, "Psst, by the way, the system attribute you see on this item? Yeah, it's a total fabrication. The real file doesn't have that attribute."

When it comes time to copy the file, the shell looks at the item ID and says, "Well, it says here that the original has the system attribute, so I'll set the system attribute on the copy." The shell copy engine doesn't know that the attribute is a lie.

This problem was fixed in Windows 8 as a side-effect of a re-write of the way the shell copy engine copies files. The shell now uses the `CopyFile2` function to copy files, relying on the kernel function to do the heavy lifting, and using the callback function to monitor progress and possibly cancel the operation. The kernel function doesn't know about these mysterious shell item IDs. All it knows how to do is copy files, and it obtains the attributes directly from the source file, which as we recall is marked hidden but not system.

Bonus chatter: The "heavy lifting" alluded to above can be quite substantial. In addition to copying the file contents, it also copies the alternate data streams and file attributes, and can also take advantage of things like copy offload.

Raymond Chen

Follow