The curse of the current directory

devblogs.microsoft.com/oldnewthing/20101109-00

November 9, 2010

Raymond Chen

The current directory is both a convenience and a curse. It's a convenience because it saves you a lot of typing and enables the use of relative paths. It's a curse because of everything else.

The root cause of this curse is that the Windows NT family of operating systems keeps open a handle to the process's current directory. (**Pre-emptive Yuhong Bao comment**: The Windows 95 series of operating systems, on the other hand, did not keep the current directory open, which had its own set of problems not relevant to this discussion.)

The primary consequence of this curse is that you can't delete a directory if it is the current directory of a running process. I see people stumble upon this all the time without realizing it.

> I am trying to delete a directory X, but when I try, I get the error message `The process cannot access the file because it is being used by another process.`. After some hunting around, I found that directory X is being held open by `someapp.exe`. Why the heck is `someapp.exe` holding my directory open, and how do I get it to stop?

The value of `someapp.exe` changes over time, but the underlying problem is the same. And when this happens, people tend to blame `someapp.exe` for stupidly holding a directory open.

Most of the time, `someapp.exe` is just a victim of the curse of the current directory.

First, let's take the case where `someapp.exe` is `explorer.exe`. Why is the current directory of Explore set to this directory?

Well, one reason might be another curse of the current directory, namely, that the current directory is a process-wide setting. If a shell extension decided to call `SetCurrentDirectory`, then that changes the current directory for all of Explorer. And if that shell extension doesn't bother to call `SetCurrentDirectory` a second time to reset the

current directory to what it was, then the current directory gets stuck at the new directory, and Explorer has now been conned into changing its current directory permanently to your directory.

Mind you, the shell extension might have tried to do the right thing by setting the current directory back to its original location, but the attempt might have failed:

```
GetCurrentDirectory(Old) // returns C:\Previous
SetCurrentDirectory(New) // changes to C:\Victim
.. do stuff ..
SetCurrentDirectory(Old) // changes to C:\Previous - fails?
```

That second call to `SetCurrentDirectory` can fail if, while the shell extension is busy doing stuff, the directory `C:\Previous` is deleted. Now the shell extension can't change the directory back, so it's left stuck at `C:\Victim`, and now you can't delete `C:\Victim` because it is Explorer's new current directory.

(The preferred behavior, by the way, is for the shell extension not to call `SetCurrentDirectory` in the first place. Just operate on full paths. Since the current directory is a process-wide setting, you can't be sure that some other thread hasn't called `SetCurrentDirectory` out from under you.)

Mind you, making the current directory a per-thread concept doesn't solve this problem completely, because the current directory for the thread (if such a thing existed) would still have a handle open until the thread exited. But if the current directory had been a per-thread concept, and if the thread were associated with an Explorer window, then closing that window would at least encourage that thread to exit and let you unstick the directory. That is, unless you did a `TerminateThread`, in which case the handle would be leaked and your attempt to release the handle only ensures that it never happens. (Note to technology hypochondriacs: This paragraph was a hypothetical and consequently will be completely ineffective at solving your problem.)

The story isn't over yet, but I'll need to digress for a bit in order to lay the groundwork for the next stage of the curse.

Bonus chatter: Hello, people. "The story isn't over yet." Please don't try to guess the next chapter in the story.

Raymond Chen

Follow