Clean-up functions can't fail because, well, how do you clean up from a failed clean-up?

devblogs.microsoft.com/oldnewthing/20080107-00

January 7, 2008

Raymond Chen

Commenter Matt asks how you're supposed to handle failures in functions like `fclose` or `CloseHandle`. Obviously, you can't. If a clean-up function fails, there's not much you can do because, well, how do you clean up from a failed clean-up? These clean-up functions fall into the category of "Must not fail for reasons beyond the program's control." If a program tries to close a file and it gets an error back, what can it do? Practically speaking, nothing. The only way a clean-up function can fail is if the program fundamentally screws up, say by attempting to close something that was never open or otherwise passing an invalid parameter. It's not like a program can try to close the handle again (or worse go into loop closing the handle repeatedly until it finally closes). Remember this when writing your own clean-up functions. Assuming the parameters are valid, a clean-up function must succeed. (I will now ruin my rhetorical flourish by yammering about the `fclose` function because if I don't, people will bring it up in the comments anyway. The `fclose` function does extra work before closing, and that extra work may indeed run into problems, but the important thing is that when `fclose` returns, the stream is well and truly closed.)

Addendum: Once again I wish to emphasize that while it may be possible for functions like `fclose` to run into errors while they are closing the stream, the point is that the result of the call to `fclose` is always a closed stream. I think most of the comments are losing sight of the point of my article and rat-holding on the various ways `fclose` can run into errors. That's not the same as *failing*. It never fails; it always succeeds, but possibly with errors.

Raymond Chen

Follow