Why do BackupRead and BackupWrite require synchronous file handles?

devblogs.microsoft.com/oldnewthing/20121221-00

December 21, 2012

Raymond Chen

The `BackupRead` and `BackupWrite` functions require that the handle you provide by synchronous. (In other words, that they *not* be opened with `FILE_FLAG_OVERLAPPED`.) A customer submitted the following question:

> We have been using asynchronous file handles with the `BackupRead`. Every so often, the call to `BackupRead` will fail, but we discovered that as a workaround, we can just retry the operation, and it will succeed the second time. This solution has been working for years.
>
> Lately, we've been seeing crash when trying to back up files, and the stack traces in the crash dumps appear to be corrupted. The issue appears to happen only on certain networks, and the problem goes away if we switch to a synchronous handle.
>
> Do you have any insight into this issue? Why were the `BackupRead` and `BackupWrite` functions designed to require synchronous handles?

The `BackupRead` and `BackupWrite` functions have historically issued I/O against the handles provided on the assumption that they are synchronous. As we saw a while ago, doing so against an asynchronous handle means that you're playing a risky game: If the I/O completes synchronously, then nobody gets hurt, but if the I/O goes asynchronous, then the temporary `OVERLAPPED` structure on the stack will be updated by the kernel when the I/O completes, which could very well be after the function that created it has already returned. The result: A stack smash. (Related: Looking at the world through kernel-colored glasses.) This oversight in the code (blindly assuming that the handle is a synchronous handle) was not detected until 10 years after the API was originally designed and implemented. During that time, backup applications managed to develop very tight dependencies on the undocumented behavior of the backup functions. The backup folks tried fixing the bug but found that it ended up introducing massive compatibility issues. On top of that, there was no real business case for extending the `BackupRead` and `BackupWrite` functions to accept asynchronous handles. As a result, there was no practical reason for changing the function's

behavior. Instead, the requirement that the handle be synchronous was added to the documentation, along with additional text explaining that if you pass an asynchronous handle, you will get “subtle errors that are very difficult to debug.”

In other words, the requirement that the handles be synchronous exists for backward compatibility.

Raymond Chen

Follow