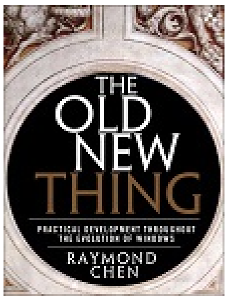
Why does Windows even have Interlocked functions when we have std::atomic?

devblogs.microsoft.com/oldnewthing/20250612-00/?p=111265

June 12, 2025



Windows provides a family of functions for performing atomic operations. They have the word `Interlocked` in their name. But why do these functions even exist when we have `std::atomic`?

This is similar to asking why they didn't use the Space Shuttle to rescue the Apollo 13 astronauts. You have the history wrong.

Multithreading and atomic operations were not added to the C and C++ languages until C11 and C++11 (respectively), but Windows was doing multithreading since Windows NT 3.1, which released in 1993. Compilers of that era typically did not include intrinsics for atomic operations, so if you needed to do things atomically, you had to call to an outside function.

The Windows `Interlocked` functions were those outside functions. Originally, the only operations were to increment, decrement, or exchange a 4-byte value. (And the increment and decrement operations didn't even tell you the result of the increment/decrement, only the sign.) Gradually, other operations were added, and the size of the operated-on value was expanded to include pointer-sized values.

In the meantime, compilers realized that maybe they should add intrinsics for atomic operations, and the C and C++ standards committees realized that atomic operations would be a nice thing to standardize, so they added `_Atomic` and `std::atomic`.

So which one should you use?

If you are coding in C or C++, you may as well use the language built-in operations. Nowadays, the compiler can inline the atomic operations, rather than taking the expense of a function call. And they also can optimize around them, or *not* optimize around them, which is probably even more important.

So why do the `Interlocked` functions still exist?

The implementations of the `Interlocked` functions are typically quite small, so leaving them around doesn't incur much of a cost. And they are still necessary if you want to perform atomic operations from languages that don't have built-in atomics.

Bonus chatter: Nowadays, the interlocked functions are typically defined for C and C++ consumers in terms of compiler intrinsics, so at the end of the day, it really doesn't matter. They all end up turning into compiler intrinsics if you are coding in C and C++.