Why do all this coroutine stuff if you can just use std::future?

devblogs.microsoft.com/oldnewthing/20211101-00

November 1, 2021

Raymond Chen

One of my colleagues reacted to my series on creating custom coroutines by asking, "Why did you bother with this `simple_task` custom coroutine? Why not just use `std::future` objects produced by `std::async`? You can `co_await` a `std::future`."

Okay, first of all, the ability to `co_await` a `std::future` is not a feature of the standard (as of this writing). It's something that the Microsoft STL provides as a courtesy. And that courtesy has not always been smooth sailing.

For another thing, `std::async` lets you pick how the lambda runs, but your only options are to run it on a separate thread, or to run the lambda only when the value is awaited. The lambda itself can't do any `co_await`ing because it is not a coroutine. It's just a plain synchronous lambda.

But the real kicker is that the Microsoft STL implementation of `std::future` is written in… PPL, The very library we are trying to get rid of by switching to coroutines in the first place.

That last bit was the clincher as far as my colleague was concerned. He sent me the astronaut with a gun meme, but instead of asking, "Wait, it's all Ohio?", the first astronaut asks, "std::future is just PPL?" The reply from the astronaut with the gun remains "Always has been."

Meme background: Wait it's all Ohio? Always has been.

Raymond Chen

Follow