Subject: Re: Coroutines package for U++
Posted by Oblivion on Mon, 07 Nov 2022 19:07:01 GMT
View Forum Message <> Reply to Message

Hello koldo,
Quote:why does a programmer need to use coroutines?

Simple: To write non-blocking APIs or programs without spaghetti code: Recall the initial version of our very own SSH package. It was designed to be fully NB. This was achieved using a blend of an event queue, storing callbacks in a bivector (in a predefined order) and tracking their state. It did work, but the underlying code was very complex. (Fun fact: The same mechanism (but a stripped down version) is still used in NetProxy package.)

Now, the coroutines eliminates BOTH. You don't need to track the state of the internals of the NB object externally or worry about a state machine. Basically it is done for you by the compiler.

You write a local code block, and mark some points (usually where the code WOULDBLOCK) to suspend the coroutine and the compiler simply suspends it and returns the control back to its caller. The suspended coroutine can be resumed later from where it is left off.

This simplifites writing NB/async apps without using threads. Not to mention that you you don't need to worry about on how to terminate it. (CoRoutines are scope bound objects.)

Hello Peter:

Quote:I do not fully understand it yet and the purpose of it.

Well I don't think this debate (stackfull/stackless, i mean) will ever end --theoretically, at least. In practice, however, stackless coroutines are selected for C++20 (as in AFAIK C# and Python).

They can't be nested directly, yes.Some see this as a disadvantage, some don't. (I'm on the latter camp, since it forces simpler coroutines. (as no nesting is possible.)

As for the GDB: Yes it still can't handle coroutines properly..

A side note: I will add more complex examples to the package along with docs...

Best regards,

Oblivion