`when_all()` is just `just()`

Document Number: P4217R0
Date: 2026-05-06
Reply-to: Robert Leahy <rleahy@rleahy.ca>
Audience: SG1, LEWG

Abstract

This paper proposes giving `std::execution::when_all()` the same meaning as `std::execution::just()`.

Background

The standard currently specifies, by fiat, that `std::execution::when_all()` is ill-formed (§33.9.12.12 [exec.when.all]):

"The expressions `when_all(sndrs...)` and `when_all_with_variant(sndrs...)` are ill-formed if any of the following is `true`:

- *`sizeof...(sndrs)` is `0`, or*
- *[...]"*

If this restriction were not present the asynchronous operation which results from connecting the result of `std::execution::when_all()` and starting the operation state yielded thereby would hang due to the fact the implementation of `std::execution::start` therefor does not complete the operation eagerly when there are no child senders (ibid.).

Discussion

`std::execution::when_all` "*adapt[s] multiple input senders into a sender that completes when all input senders have completed*" (ibid.). Given zero senders "all input senders" have always trivially completed (in the same way that `std::all_of` returns `true` for empty input) and therefore there's no reason to ban `std::execution::when_all()`, it is simply equivalent to `std::execution::just()`.

Banning `std::execution::when_all()` (i.e. the status quo) unnecessarily creates a special case when writing generic algorithms.

Proposal

[exec.when.all]

[...]

The names `when_all` and `when_all_with_variant` denote customization point objects. Let `sndrs` be a pack of subexpressions and let `Sndrs` be a pack of the types `decltype((sndrs))...`. The expressions `when_all(sndrs...)` and `when_all_with_variant(sndrs...)` are ill-formed if ~~any of the following is `true`:~~

- ~~`sizeof...(sndrs)` is `0`, or~~
- `(sender<Sndrs> && ...)` is `false`.

The expression `when_all(sndrs...)` is expression-equivalent to:

- `make-sender(when_all, {}, sndrs...)` if `sizeof...(sndrs)` is not `0`, or
- `just()` otherwise.

[...]

Acknowledgements

The author would like to thank Jonathan Müller for his C++Now 2026 talk which made him aware of this defect.