Specifying Trivially Relocatable Types in the Standard Library

Adding a new specification element for class properties

Document #: DxxxxR0
Date: 2024-MM-DD
Project: Programming Language C++
Audience: Library Evolution
Reply-to: Alisdair Meredith
<ameredith1@bloomberg.net>

Contents

1 Abstract

[P2786R5] introduces the notion of trivial relocatability to the C++ Standard without making specific recommendations on which parts of the Standard Library should provide guarantees regarding the new facility. This paper will review the whole C++ Standard Library making those recommendations for which types must be trivially relocatable and which types may support such relocatability as part of their Quality of Implementation (QoI) concerns.

A new Library specification element for class properties is introduced to consistently address similar concerns across the whole library, that are treated in a more ad-hoc manner today.

2 Revision history

PRE-PRINT — THIS DOCUMENT WILL BE FINALIZED FOR THE PRE-ST LOUIS 2024 MAILING

R0 January 2024 (midterm mailing)

Initial draft of this paper.

Review status of clauses:

16 [library]	In progress
17 [support]	Not Started
18 [concepts]	Not Started
19 [diagnostics]	Not Started
20 [mem]	Not Started
21 [meta]	Not Started
22 [utilities]	Not Started
23 [strings]	Not Started
24 [containers]	Not Started
25 [iterators]	Not Started
26 [ranges]	Not Started
27 [algorithms]	Not Started
28 [numerics]	Not Started
29 [time]	Not Started
30 [localization]	Not Started
31 [input.output]	Not Started
32 [re]	Not Started
33 [thread]	Not Started
D [depr]	Not Started

3 Introduction

4 Analysis

4.1 History

5 Design Principles

6 Proposed Solution

We propose adding a new specification element, *Class properties*, for any specification related to class properties 11.2 [class.prop]. The Standard Library already makes some effort to specify whether a class must be trivially copyable, standard layout, etc., and we believe it would be more maintainable to track such specification with a consistent presentation, using a consistent form.

Once we have a *Class properties* element, we can then review all library classes and decide whether to specify the trivial relocatability behavior for that class, which might be conditional on its template arguments if it is a class template. We might also deliberately defer specifying behavior to allow for implementations making different choices, such as node-based containers allocating their end node vs storing the pointers in the container's object representation.

To avoid burdening the text with too much information, we also suggest blanket wording that says any Standard Library type that is trivially copyable is also trivially relocatable, so we do not need to repeat information that only a deliberately hostile implementation would exploit.

Finally, once we have an easy way to document class properties, we might consider making stronger guarantees on existing library components where such specification would be useful, for example clarifying which types are implicit lifetime.

6.1 Which class properties should be specified

— Trivially copyable
— Trivially relocatable
— Standard Layout
— Implicit lifetime
— Structural 13.2 [temp.param]
— Aggregate (might be just std::array?)
— Empty (likely extending current spec; relevant for many standard class templates)
— Bitmask (Library qualification, but used frequently enough)

6.1.1 Consider

— Literal types
— Layout compatible types (if the situation arises?)

6.1.2 Other

— Callable
— Final
— Integer-class
— Polymorphic

6.2 Clauses that should note interaction with class properties

— 16.3.2.4 [structure.specifications] — class properties as well as invariants
— 16.3.3.3.5 [customization.point.object] — may be mildly reformulated with the new specification element
— 16.3.3.5 [objects.within.classes] — we may be constraining which members may be added

7 Wording

Make the following changes to the C++ Working Draft. All wording is relative to [N4981], the latest draft at the time of writing.

8 Acknowledgements

Thanks to Michael Parks for the pandoc-based framework used to transform this document's source from Markdown.

9 References

[N4981] Thomas Köppe. 2024-04-16. Working Draft, Programming Languages — C++.
https://wg21.link/n4981

[P2786R5] Mungo Gill, Alisdair Meredith. 2024-04-09. Trivial Relocatability For C++26.
https://wg21.link/p2786r5