Subject: Re: How to mark std::array<T, N> moveable if only T is moveable Posted by busiek on Tue, 23 Jul 2024 16:25:28 GMT

View Forum Message <> Reply to Message

mirek wrote on Thu, 18 July 2024 08:41Lance wrote on Wed, 03 July 2024 03:39Reminds me of some exploration I did with Moveable.

I vaguely recalled that when I tested a few years back, the way Moveable was designed were causing a compile time error with Windows/VSBuild. I would also welcome a redesign of Moveable using more recent language facilities.

Suggestions? (But it needs to be backward compatible).

```
I was thinking about something like that:
#include <Core/Core.h>
#include <type_traits>
namespace Upp {
// First way of marking class moveable is to be an ancestor of NtlMoveableBase
template <class T> class NtlMoveableBase {};
// Second way is to specialize NtlMoveableClass
template <class T>
struct NtlMoveableClass
: std::integral_constant<bool,
  std::is_trivial_v<T>
|| std::is_base_of_v<Upp::NtlMoveableBase<T>, T>
// below is not needed if we make them derived from NtlMoveableBase
|| std::is base of v<Upp::Moveable <T>, T>
|| std::is_base_of_v<Upp::Moveable<T>, T>
|| std::is base of v<Upp::MoveableAndDeepCopyOption<T>, T>> {};
// For instance mark std::array<T, N> moveable if only T is moveable
template <class T, size_t N>
struct NtlMoveableClass<std::array<T, N>>
: std::integral constant<bool, NtlMoveableClass<T>::value> {}:
// Helper for checking whether class is moveable
template <class T>
inline constexpr bool IsNtlMoveable = NtlMoveableClass<T>::value;
// Optional concept for c++20
template <class T>
concept NtlMoveable = IsNtlMoveable<T>;
}
using namespace Upp;
```

```
// use of concept
template <NtlMoveable T>
struct MyVector
{
T *ptr;
// Or use static_assert
~MyVector() { static_assert(IsNtlMoveable<T>); }
CONSOLE_APP_MAIN
// Checking if given type is moveable
static_assert(IsNtIMoveable<int>);
static_assert(IsNtlMoveable<const void *>);
static_assert(IsNtlMoveable<Vector<int>>);
static assert(IsNtlMoveable<std::array<int, 5>>);
static_assert(IsNtlMoveable<std::array<Vector<int>, 5>>);
static_assert(!IsNtlMoveable<Thread>);
You could redefine NTL_MOVEABLE macro as a specialization of NtlMoveableClass.
But you need to change an assertion AssertMoveable((T*)0) to static assert(IsNtlMoveable<T>).
Are those ideas usable?
```