Subject: Re: How to mark std::array<T, N> moveable if only T is moveable
Posted by mirek on Mon, 29 Jul 2024 06:47:50 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 25 July 2024 09:32mirek wrote on Thu, 25 July 2024 09:19
Another thing to consider: Tuple is moveable if all its components are moveable. Are we able to express that somehow? (If not, no big deal, I can make Tuple moveable and request that elements are).

Figured it out:

```
struct not_moveable {};

template <class A, class B>
class Two : std::conditional<is_moveable<A> && is_moveable<B>, moveable<Two<A, B>>,
not_moveable>::type {
 A a;
 B b;
};
```

Nicer and simpler way:

```
template <class A, class B> inline constexpr bool is_moveable<Two<A, B>> = is_moveable<A>
&& is_moveable<B>;
```

I am about ready to try to introduce this into U++, but I am actually out of ideas how to name things....

Moveable / is_moveable is not perfect (future c++ might name this trivially_relocatable, but that is too long IMO), but u++ traditional, I guess I can live that.

However, new system should introduce a trait that says "I know I cannot memmove this type, but I still would like to store it into Vector anyway". E.g. to have Index<std::string> ... Really not sure what id should I use for that... Something like "is_vector_compatible_but_slower", or "std_moveable" or something?

(alternative would be to have "mem moveable" as option, but I want to force developer to be explicit, not to forget to mark moveable types)

Mirek