Subject: Re: Refactoring Moveable
Posted by mirek on Sun, 08 Sep 2024 13:41:53 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Sun, 08 September 2024 13:18Hello Mirek,

Good to be on C++17

Hovewer, this seems to break a lot of things.

For example, If I derive something from MoveableAndDeepCopyOption<T>, which is now derived from TriviallyRelocatable<T> (Say, T = Vector<T>, which was possible up until now) then I can't access the methods or members of T.

Reason: TriviallyRelocatable<T> is defined as:

```
template <class T>
struct TriviallyRelocatable {};
```

Any ideas on how to proceed, or am I missing something?

Best regards,
Oblivion

Uhm, normal use is like

```
struct Foo : MoveableAndDeepCopyOption<Foo> {
...
};
```

- obviously, you can access methods of Foo in Foo...

Example of what you need?

Note: There is one small issue I was unable to solve. U++ had two parameter Moveable, where second parameter was optional base class. It is supposed to help with MSC++ big with empty base class optimisations. It does not seem possible to use template magic with that which would go well MSC++ optimiser, putting Moveable first in the base class list seems to work fine wrt MSC++ optimisation and it really was used very sparsely even in U++ code and I guess almost never in client code.

Anyway

struct Foo : Moveable<Foo, FooBase> ...

now has to be rewritten as

struct Foo : Moveable<Foo>, FooBase ...