How do I make an expression non-movable? What's the opposite of std::move?

devblogs.microsoft.com/oldnewthing/20240306-00

March 6, 2024

Raymond Chen

You can use `std::move` to indicate that a value can be moved. But what can you do to indicate that something movable cannot be moved after all?

For example, maybe you want to force the copy assignment operator to be used instead of the move assignment operator:

```
extern Thing GetThing();

Thing thing;

// this uses the move assignment operator
thing = GetThing();
```

You might want to force the copy assignment operator because the copy assignment operator comes with additional statistics or bookkeeping. How do you make something non-movable?

You can do it by const-ifying the thing, so that the only available option is to copy it.

```
template<typename T>
std::remove_reference_t<T> const& no_move(T&& t)
{
    return t;
}
```

The `no_move` function takes any kind of reference, and then returns a `const` reference to that referred-to thing.

```
extern std::vector<int> make_vector();

// Force copy assignment, not move assignment.
v = no_move(make_vector());
```

Bonus chatter: Note that the following similar-looking code is not a move operation:

```
std::vector<int> v = make_vector();
```

The above code does not construct `v` by moving the result of `make_vector`. Rather, the above code uses copy elision, and the return value of `make_vector` is placed directly into `v` without any copying or moving at all.

But the same trick can be used to suppress copy elision.

```
// Force copy constructor, not move constructor copy elision.
std::vector<int> v = no_move(make_vector());
```