Intro To C++ Allocators

A 30 minute prelude to open discussion, questions, and answers

Louis “Luigi” Langholtz, Organizer of the NM C++ Programmers Meetup Group, 8/22/2023

Caveats: opinion, experience, mistakes?

C++ allocator?

What's that?

- Class template that satisfies the Allocator named requirement.
- Contextually synonymous with object of such a class.
- Interface to allocation/deallocation functionality from a memory resource.
- Interface for some standard library types. Examples?
- Potential interface for any storage allocating type.
- Equal to another object interfacing a compatible memory resource.

Example

Minimal C++ Allocator

From item 11 of section 16.5.3.5 Cpp17Allocator requirements of n4861.pdf…

```
template<class Tp>
struct SimpleAllocator {
    typedef Tp value_type;
    SimpleAllocator(ctor args);

    template<class T> SimpleAllocator(const SimpleAllocator<T>& other);

    [[nodiscard]] Tp* allocate(std::size_t n);
    void deallocate(Tp* p, std::size_t n);
};

template<class T, class U>
bool operator==(const SimpleAllocator<T>&, const SimpleAllocator<U>&);
template<class T, class U>
bool operator!=(const SimpleAllocator<T>&, const SimpleAllocator<U>&);
```

Categories?

“Stateless” and “Stateful”

- Allocators using static state = “stateless”.

```
std::vector<int, std::allocator<int>> my_ints; // Like std::vector<int> my_ints!
```

- Allocators with non-static data member(s) = “stateful”.

```
std::pmr::synchronized_pool_resource resource;

std::vector<int, std::pmr::polymorphic_allocator<int>> my_ints(&resource);
```

References

Going into details…

- Minimal allocator.
- Stateless vs. stateful allocators.
- ThreadLocalAllocator.
- StatsResource.