Subject: Re: Doubt with Buffer<> of a trivially destructible type
Posted by Lance on Sat, 14 Dec 2024 18:19:10 GMT
View Forum Message <> Reply to Message

This is a valid use case, and something need to be addressed in u++ libary directly instead of circling around.

Here is a simple utility we can use to fix it from with u++ libary

```
template <typename T, std::size_t...>
constexpr auto object_count(T& t)
{
 return 1u;
}

template <std::size_t ... Ns, typename T, std::size_t n>
constexpr auto object_count(T (&arr)[n] )
{
 return n * object_count(arr[0]);
}

   // eg, with
    double d;
    double a1[5];
    double a2[5][3];
    double a3[5][4][2];

    // then
    static_assert(object_count(d)==1,"?");
    static_assert(object_count(a1)==5,"?");
    static_assert(object_count(a2)==15,"?");
    static_assert(object_count(d3)==40,"?);
```

With above utility, we can easily modify u++ Vector to accomodate c style array.

basially, if T is trivially relocatible, then
any c array of T is alos trivially relocatible,
Upp::Vector don't care any detail of c array, except the total number of T objects in the array to properly construct, move and destruct it, with T::~T() properly defined of course.

Oh, for

```
     // some type T
     T d3[3][2][5];
```

a simple

```
sizeof(d3)/sizeof(T)
```

will do the job :)