Mundane std::tuple tricks: Selecting via an index sequence

devblogs.microsoft.com/oldnewthing/20200623-00

June 23, 2020

Raymond Chen

Last time, we combined tuples. That's relatively straightforward. Splitting them apart is harder.

The `std::index_sequence` is a standard type which captures a sequence of zero or more nonnegative integers into a type. It's a special case of the `std::integer_sequence` which captures integer values of user-specified type, where the user-specified type is a `std::size_t`.

Tuple splitting boils down to a fold expression involving `std::index_sequence`.

```
// Don't use this; see discussion.
template<typename Tuple, std::size_t... Ints>
auto select_tuple(Tuple&& tuple, std::index_sequence<Ints...>)
{
 return std::make_tuple(
    std::get<Ints>(std::forward<Tuple>(tuple))...);
}
```

This is the heart of tuple splitting, so let's take it apart.

The first template type parameter is the tuple being manipulated. It is captured as a universal reference so that we can forward it. This preserves rvalue-ness, which is particularly important in case some of the types in the tuple are move-only. (It also helps if the types are both copyable and movable, because it will prefer the move, which is usually much less expensive than the copy.)

The remaining template parameters are `size_t` values, representing the indices in the `index_sequence`.

The fold expression is

```
    (std::get<Ints>(std::forward<Tuple>(tuple))...)
```

which forms the parameter list to `make_tuple`. The expression is repeated once for each value in the `Ints...` parameter pack, resulting in a series of parameters which pluck the corresponding indexed values from the tuple.

For example,

```
auto result = select_tuple(
    std::make_tuple('x', 3.14, 'z'),
    std::index_sequence<2, 1, 1, 2>{});
```

We take a three-element tuple `('x', 3.14, 'z')` and select from it the index sequence `<2, 1, 1, 2>`. The fold expression becomes

```
    (std::get<2>(std::forward<Tuple>(tuple)),
     std::get<1>(std::forward<Tuple>(tuple)),
     std::get<1>(std::forward<Tuple>(tuple)),
     std::get<2>(std::forward<Tuple>(tuple)))
```

This extracts items 2, 1, 1, and 2 from the tuple, passing them to `make_tuple`, which recombines them into the resulting tuple `('z', 3.14, 3,14, 'z')`. Note that indices 1 and 2 were extracted multiple times, and index 0 was not extracted at all. Note also that the size of the resulting tuple matches the number of indices, not the size of the source tuple.

Note that if the values in the tuple had been a movable type (not to be confused with movable type), then extracting indices `<2, 1, 1, 2>` would have resulted in moving some of the items more than once. That's probably not going to produce a happy result, so you don't usually extract a value more than once. (Though there's nothing stopping you.)

There's a defect with our `select_ tuple` function, though. I alluded to it in the comment. We'll address that defect next time.

Raymond Chen

Follow