Mundane std::tuple tricks: Getting started

devblogs.microsoft.com/oldnewthing/20200622-00

June 22, 2020

Raymond Chen

The C++ standard library `tuple` is quite versatile. It's a handy way of grabbing a bunch of types or values into a single unit, and the C++ standard library also provides a number of helpers to manipulate them.

For example, `make_tuple` lets you manufacture a tuple from values, which is handy if you want to capture a template parameter pack into something you can manipulate.

```
[](auto... args)
{
 auto args_tuple = std::make_tuple(std::move(args)...);
}
```

We learned earlier that `std::tuple_element_t` lets you pluck a single type out of a tuple, and `std::get` lets you extract a single value.

And then there's `tuple_cat` which concatenates two tuples. Though it concatenates values, not types. But writing a version that concatenates types isn't hard.

```
template<typename T1, typename T2> struct tuple_cat_helper;
template<typename... T1, typename...T2>
struct tuple_cat_helper<std::tuple<T1...>, std::tuple<T2...>>
{
    using type = std::tuple<T1..., T2...>;
};

template<typename T1, typename T2>
using tuple_cat_t = typename tuple_cat_helper<T1, T2>::type;

// example is std::tuple<int, char, double>
using example = tuple_cat_t<std::tuple<int>,
                            std::tuple<char, double>>;
```

We define a templated `tuple_cat_helper` with a specialization that sucks out the tuple types and generates a new tuple whose types are the concatenation of the two type lists. And then we provide a type alias that reaches in and grabs the `type` dependent type.

Or you can be sneaky and let the existing `tuple_cat` do the heavy lifting:

```
template<typename T1, typename T2>
using tuple_cat_t = decltype(std::tuple_cat(std::declval<T1>(),
                                            std::declval<T2>()));
```

And since `tuple_cat` can concatenate multiple tuples, we can write

```
template<typename... Tuples>
using tuple_cat_t = decltype(std::tuple_cat(std::declval<Tuples>()...));
```

Getting more while doing less.

This is all great for putting tuples together, but there's nothing in the standard library for taking tuples apart.

We'll start that next time.

Bonus chatter: I wasn't quite telling the truth when I said that `make_tuple` can capture a template parameter pack. We'll come back to this issue later.

Raymond Chen

Follow