Hiding C++ template parameter packs in a tuple

devblogs.microsoft.com/oldnewthing/20200529-00

May 29, 2020

Raymond Chen

C++11 introduced variadic templates and template parameter packs.

```
template<typename... Args>
struct S;
```

Passing template parameter packs around is a bit of a hassle, because the dots "soak up" parameters, which make them hard to pass to other templated types. You also can't "save" parameter packs in another type:

```
template<typename... Args>
struct S
{
    // doesn't compile: cannot create a type that
    // is itself a parameter pack
    using TheArgs = Args...;
};
```

One workaround for this is to capture the types into a `std::tuple`.

```
template<typename... Args>
struct S
{
    using Tuple = std::tuple<Args...>;
};
```

You can then pass the `Tuple` around, and if you need to extract the types, you can pull them out of the tuple.

My first thought about how to extract the types was to use `std::get`:

```
// code in italics is wrong
template<typename... Args>
struct Traits
{
    using Tuple = std::tuple<Args...>;
    static constexpr auto Size = sizeof...(Args);
    template <std::size_t N>
    using Nth = decltype(std::get<N>(std::declval<Tuple>()));
    using First = Nth<0>;
    using Last  = Nth<Size - 1>;
};
```

This doesn't work because `std::get` returns a reference:

```
void f()
{
 whatis<Traits<int, double>::First>();
}

    call    void whatis<int&&>()
```

This behavior is presumably so you can modify the tuple:

```
std::tuple<int> tuple;
std::get<0>(tuple) = 2;
```

Fortunately, there's another way to extract the type from a tuple: `std::tuple_element`.

```
template<typename... Args>
struct Traits
{
    using Tuple = std::tuple<Args...>;
    static constexpr auto Size = sizeof...(Args);
    template <std::size_t N>
    using Nth = typename std::tuple_element<N, Tuple>::type;
    using First = Nth<0>;
    using Last  = Nth<Size - 1>;
};
```

This provides a simpler way to extract the last type from a template parameter pack than writing some horrible recursive template metaprogram, which is what some people did instead of hiding the pack inside a tuple.

Raymond Chen

Follow