Pulling a single item from a C++ parameter pack by its index

devblogs.microsoft.com/oldnewthing/20240516-00

May 16, 2024

Raymond Chen

Suppose you have a C++ parameter pack and you want to pluck out an item from it by index.

```
template<int index, typename...Args>
void example(Args&&... args)
{
    // how do I access the index'th args parameter?
}
```

One solution is to use `std::tie`:

```
template<int index, typename...Args>
void example(Args&&... args)
{
    auto& arg = std::get<index>(
        std::tie(args...));
}
```

We expand the parameter pack into the parameters of `std::tie`, which returns a tuple of lvalue references. We can then pull out the element at the specified index via `std::get<0>()`.

The catch here is that `std::tie` always captures the references as lvalue references. If you want to preserve the reference category of the original parameters, you can use `std::forward_as_tuple`:

```
template<int index, typename...Args>
void example(Args&&... args)
{
    auto&& arg = std::get<index>(
        std::forward_as_tuple(
            std::forward<Args>(args)...));
}
```

This time, we apply `std::forward` to each of the elements of the parameter pack (to preserve the original reference category), then pass them all as parameters to `std::forward_as_tuple`. Unlike `std::tie`, `std::forward_as_tuple` preserves the reference categories of its arguments,

so it produces a tuple of lvalue and rvalue references that match the original parameters. Again, we pull out the element via `std::get<index>()`, and capture it as an `auto&&` reference, which again preserves the reference category.

For future reference, you can capture the `Args` at a specific index by extracting it from the tuple, or more easily, getting it from the variable.

```
template<int index, typename...Args>
void example(Args&&... args)
{
    auto&& arg = std::get<index>(
        std::forward_as_tuple(
            std::forward<Args>(args)...));

    // The hard way
    using Arg = std::tuple_element_t<index,
        std::tuple<Args&&...>>;

    // The lazy way
    using Arg = decltype(arg);
}
```

Bonus chatter: There is a C++26 proposal to add core language support for pulling elements out of a parameter pack by index: Pack Indexing. Under the proposed syntax, the above code would be simplified to

```
template<int index, typename...Args>
void example(Args&&... args)
{
    // Extract as lvalue
    auto& arg = args...[index];
    using Arg = Args...[index];

    // Preserve reference category
    auto&& arg = (Args...[index]&&)args...[index];
    using Arg = Args...[index]&&;
}
```