A trick for keeping an object alive in a C++ lambda while still being able to use the this keyword to refer to it

devblogs.microsoft.com/oldnewthing/20190104-00

January 4, 2019

Raymond Chen

You may want to capture your `this` pointer into a C++ lambda, but that captures the raw pointer. If you need to extend the object's lifetime, you will need to capture a strong reference. For plain C++ code, this would be a `std::shared_ptr`. For COM objects, this is usually some sort of smart pointer class like `ATL::CComPtr`, `Microsoft::WRL::ComPtr`, or `winrt::com_ptr`.

```
// std::shared_ptr
auto callback = [self = shared_from_this()]() {
  self->DoSomething(self->m_value);
  self->DoSomethingElse();
 };

// WRL::ComPtr
auto callback = [self =
                 Microsoft::WRL::ComPtr<ThisClass>(this)]() {
  self->DoSomething(self->m_value);
  self->DoSomethingElse();
};

// ATL::CComPtr
auto callback = [self =
                 ATL::CComPtr<ThisClass>(this)]() {
  self->DoSomething(self->m_value);
  self->DoSomethingElse();
};

// winrt::com_ptr
template<typename T>
auto to_com_ptr(T* p) noexcept
{
  winrt::com_ptr<T> ptr;
  ptr.copy_from(p);
  return ptr;
}

auto callback = [self = to_com_ptr(this)] {
  self->DoSomething(self->m_value);
  self->DoSomethingElse();
};
```

A common pattern for the “capture a strong reference to yourself” is to capture both a strong reference and a raw `this`. The strong reference keeps the `this` alive, and you use the `this` for convenient access to members.

```
// std::shared_ptr
auto callback = [lifetime = std::shared_from_this(this),
                 this]() {
  DoSomething(m_value);  // was self->DoSomething(self->m_value);
  DoSomethingElse();     // was self->DoSomethingElse();
 };

// WRL::ComPtr
auto callback = [lifetime =
                 Microsoft::WRL::ComPtr<ThisClass>(this),
                 this]() {
  DoSomething(m_value);  // was self->DoSomething(self->m_value);
  DoSomethingElse();     // was self->DoSomethingElse();
};

// ATL::CComPtr
auto callback = [lifetime =
                 ATL::CComPtr<ThisClass>(this),
                 this]() {
  DoSomething(m_value);  // was self->DoSomething(self->m_value);
  DoSomethingElse();     // was self->DoSomethingElse();
};

// winrt::com_ptr
auto callback = [lifetime = to_com_ptr(this),
                 this]() {
  DoSomething(m_value);  // was self->DoSomething(self->m_value);
  DoSomethingElse();     // was self->DoSomethingElse();
};
```

I like to give the captured strong reference a name like `lifetime` to emphasize that its purpose is to extend the lifetime of the `this` pointer. Otherwise, somebody might be tempted to “optimize” out the seemingly-unused variable.

Raymond Chen

Follow