Subject: How to use callback with 3 parameters
Posted by malya on Thu, 10 Jan 2008 11:35:31 GMT
View Forum Message <> Reply to Message

```
void MyFunc(int i, int i1, int i2)
{
   ...
}

void main()
{
   PostCallback(callback(MyFunc, 1, 2, 3)); // ?????
}
```

Subject: Re: How to use callback with 3 parameters
Posted by mirek on Thu, 10 Jan 2008 16:38:30 GMT
View Forum Message <> Reply to Message

malya wrote on Thu, 10 January 2008 06:35

```
void MyFunc(int i, int i1, int i2)
{
   ...
}

void main()
{
   PostCallback(callback(MyFunc, 1, 2, 3)); // ?????
}
```

PostCallback(callback3(MyFunc, 1, 2, 3)); // ?????

Subject: Re: How to use callback with 3 parameters
Posted by malya on Thu, 10 Jan 2008 19:30:43 GMT
View Forum Message <> Reply to Message

Quote:
PostCallback(callback3(MyFunc, 1, 2, 3)); // ?????

There is an error:

error C3861: 'callback3': identifier not found, even with argument-dependent lookup

Subject: Re: How to use callback with 3 parameters
Posted by malya on Fri, 11 Jan 2008 08:47:38 GMT
View Forum Message <> Reply to Message

Please post me a small example.
Thanks!

Subject: Re: How to use callback with 3 parameters
Posted by mrjt on Fri, 11 Jan 2008 09:17:36 GMT
View Forum Message <> Reply to Message

callback3 does not exits at the moment, the closest is callback2. You will have to use another technique, such as packing the data into a structure and passing that instead.

```
struct int3
{
 int3(int _a, int _b, int _c) : a(_a), b(_b), c(_c) { }
 int a, b, c;
};

void MyFunc(int3 i)
{
    ...
}

void main()
{
    PostCallback(callback1(MyFunc, int3(1, 2, 3))); // ?????
}
```

There are other ways around this also.

Subject: Re: How to use callback with 3 parameters
Posted by mirek on Fri, 11 Jan 2008 10:11:18 GMT
View Forum Message <> Reply to Message

malya wrote on Fri, 11 January 2008 03:47Please post me a small example.
Thanks!

Suprisingly, callback3 was somehow forgotten, while callback4 is there...

Well, it is now fixed. Quick fix

```
template <class OBJECT_, class METHOD_, class T1, class T2, class T3>
struct CallbackMethodActionArg3Pte : public CallbackAction {
 Ptr<OBJECT_>  object;
 METHOD_       method;
 T1            arg1;
 T2            arg2;
 T3            arg3;
 void    Execute() { if(object) (object->*method)(arg1, arg2, arg3); }

 CallbackMethodActionArg3Pte(OBJECT_ *object, METHOD_ method, T1 arg1, T2 arg2, T3 arg3)
 : object(object), method(method), arg1(arg1), arg2(arg2), arg3(arg3) {}
};

template <class Object, class R, class O, class A, class B,class C, class T1, class T2, class T3>
Callback pteback3(Object *object, R (O::*method)(A, B, C), T1 arg1, T2 arg2, T3 arg3) {
 return Callback(new CallbackMethodActionArg3Pte<Object, R (O::*)(A,B,C), T1, T2, T3>
                 (object, method, arg1, arg2, arg3));
}

template <class OBJECT_, class METHOD_, class T1, class T2, class T3>
struct CallbackMethodActionArg3 : public CallbackAction
{
 OBJECT_  *object;
 METHOD_   method;
 T1        arg1;
 T2        arg2;
 T3        arg3;

 void    Execute() { (object->*method)(arg1, arg2, arg3); }

 CallbackMethodActionArg3(OBJECT_ *object, METHOD_ method, T1 arg1, T2 arg2, T3 arg3)
 : object(object), method(method), arg1(arg1), arg2(arg2), arg3(arg3) {}
};

template <class Object, class R, class O, class A, class B, class C, class T1, class T2, class T3>
Callback callback3(Object *object, R (O::*method)(A, B, C), T1 arg1, T2 arg2, T3 arg3)
{
 return Callback(
  new CallbackMethodActionArg3<Object, R (O::*)(A, B, C), T1, T2, T3>(object, method, arg1,
arg2, arg3));
}

template <class Object, class R, class O, class A, class B, class C, class T1, class T2, class T3>
Callback callback3(const Object *object, R (O::*method)(A, B, C) const, T1 arg1, T2 arg2, T3
arg3) {
 return Callback(new CallbackMethodActionArg4<Object, R (O::*)(A, B, C) const, T1, T2, T3>
                 (object, method, arg1, arg2, arg3));
```

```
}

template <class X, class T1, class T2, class T3, class HC = X>
struct CallbackActionCallArg3 : public CallbackAction {
 X        x;
 T1       arg1;
 T2       arg2;
 T3       arg3;
 void    Execute() { x(arg1, arg2, arg3); }

 CallbackActionCallArg3(X x, T1 arg1, T2 arg2, T3 arg3)
  : x(x), arg1(arg1), arg2(arg2), arg3(arg3) {}
};

template <class R, class A, class B, class C, class T1, class T2, class T3>
Callback callback3(R (*fn)(A, B, C), T1 arg1, T2 arg2, T3 arg3) {
 return Callback(
  new CallbackActionCallArg3<R (*)(A, B, C), T1, T2, T3, uintptr_t>(fn, arg1, arg2, arg3));
}

template <class A, class B, class C, class T1, class T2, class T3>
Callback callback3(Callback3<A, B, C> cb, T1 arg1, T2 arg2, T3 arg3) {
 return Callback(
  new CallbackActionCallArg3<Callback4<A, B,C,D>, T1, T2, T3>(cb, arg1, arg2, arg3));
}
```

add to Core/Callback.h.

Note: In fact, the reason why this was unnoticed until now is that for "higher order callbacks", it is usually better to pass the struct as single parameter. We were reluctant to extend the number of parameters beyond 2, which is the maximum that was ever needed for CtrlLib or any application we did.

Mirek

Subject: Re: How to use callback with 3 parameters
Posted by malya on Fri, 11 Jan 2008 13:11:42 GMT
View Forum Message <> Reply to Message

Thanks for quick reply!
