Subject: Callback (THISBACK) Improve
Posted by tojocky on Tue, 07 Dec 2010 14:08:52 GMT
View Forum Message <> Reply to Message

Hello All,

I propose to impove a little callback mechanism by adding new functionality.

Example: In a class I have

Callback2<P1, P2, P2, P3, P4> when_callback;

From other class I want to attach a nethod

```
OnCallback1(P1 p1, P2 p2, P3 p3, P4 p4, T1 t1, T2 t2, T3 t3){
    ...
}
```

When I try to attach:

Attach(){when_callback = THISBACK3(OnCallback1, val1, val2, val3)};

Returned an error.

With the attached patch this functionality is integrated.
Maximum can be 4 parameters and 4 arguments(constants). Total parameters can be 8.

Maybe I give a non-simple example.

If this functionality will be accepted that I can improve a little with functions(not methods).

A little example is:

```
#include <Core/Core.h>

using namespace Upp;

struct Foo {
 int x;

 void Action()               { Cout() << "Action: " << x << '\n'; }
 void ActionWithParam(int y)  { Cout() << "ActionWithParam: " << x + y << '\n'; }

 Callback WhenDo;
 Callback1<int> WhenDo1;
```

```
 Callback4<int, String, String, String> WhenDo4;
 void Do()                  { WhenDo(); WhenDo1(2);WhenDo4(4, "2 par", "3 par", "4 par");}

 Foo(int x = 0) : x(x) {}
};

void Fn()
{
 Cout() << "Fn!" << '\n';
}

struct Bar {
 Foo foo;

 void Action() { Cout() << "foo's Do called\n"; }
 void ActionMaxPar(int p1, String p2, String p3, String p4) { Cout() << "foo's Do called with max
par par 4 = "<<p4<<"\n"; }
 void ActionMaxParAndArg(int p1, String p2, String p3, String p4, int a1, String a2, String a3,
String a4) { Cout() << "foo's Do called with max par and arg arg4="<<a4<<"\n"; }

 typedef Bar CLASSNAME;

 Bar() { foo.WhenDo = THISBACK(Action);
  String v_arg1("1 arg");
  String v_arg2("2 arg");
  String v_arg3("3 arg");
  String v_arg4("4 arg");
  int v_arg1_i = 1;

  foo.WhenDo1 = THISBACK3(ActionMaxPar, v_arg1, v_arg2, v_arg3);
  foo.WhenDo4 = THISBACK(ActionMaxPar);
  foo.WhenDo4 << THISBACK4(ActionMaxParAndArg, v_arg1_i, v_arg2, v_arg3, v_arg4);
 }
};

struct Safe : Pte<Safe> {
 void Action() { Cout() << "safe action!\n"; }
};

CONSOLE_APP_MAIN
{
 Foo a(10);
 Callback cb1 = callback(&a, &Foo::Action);
 Callback cb2 = callback(Fn);
 Callback1<int> cb3 = callback(&a, &Foo::ActionWithParam);
 Callback cb4 = callback1(&a, &Foo::ActionWithParam, 30);

 cb1();
```

```
 cb2();
 cb3(10);
 cb4();

 Cout() << "---------\n";
 cb4 << cb2;
 cb4();

 Cout() << "---------\n";
 Bar b;
 b.foo.Do();

 Cout() << "---------\n";
 {
  Safe f;
  cb4 = pteback(&f, &Safe::Action);
  Cout() << "callback valid: " << (bool)cb4 << '\n';
  cb4();
 }
 Cout() << "callback valid: " << (bool)cb4 << '\n';
 cb4();
}
```

(example attached)

Ion Lupascu (tojocky)

File Attachments

1) Callback.h, downloaded 437 times
2) Callback_example.zip, downloaded 378 times

Subject: Re: Callback (THISBACK) Improve
Posted by koldo on Thu, 09 Dec 2010 08:08:06 GMT
View Forum Message <> Reply to Message

Hello Ion

Is what you are proposing a kind of variable argument list callback?

Subject: Re: Callback (THISBACK) Improve
Posted by tojocky on Thu, 09 Dec 2010 12:42:31 GMT
View Forum Message <> Reply to Message

koldo wrote on Thu, 09 December 2010 10:08Hello Ion

Is what you are proposing a kind of variable argument list callback?
Hello Koldo,

Yes, you are right.
I propose to add possibility to add variable argument list into callback. It was implemented only for Callback1 class and can pas only one variable arguments. Example:

```
Class a {
  typedef a THISBACK;
  Callback<int> WenDO;
}
Class b{
  typedef b THISBACK;
  a a_val1;
  void OnDo(int, int);
  void OnDo3(int, int);
  void Initialize(){a_val1.WhenDo = THISBACK1(OnDo, 3);}
}
```

Exists situation when I want to pass 2 variable arguments to callback class value (a_val1.WhenDo << THISBACK2(OnDo3, 3, 2).
Maximum variable argument list is 4 for all 4 Callback class templates (Callback1, Callback2, Callback3, Callback4).

Thank you for persons who introduced the term of callback.

For me it is very useful and seems to be very simple.

Subject: Re: Callback (THISBACK) Improve
Posted by mirek on Sun, 12 Dec 2010 07:43:10 GMT
View Forum Message <> Reply to Message

So this is basically adding more combinations of arguments and parameters to Callback, similar to what was alredy there with callback1, right?

OK, why not. Patch applied.

Mirek

Subject: Re: Callback (THISBACK) Improve
Posted by tojocky on Sun, 12 Dec 2010 12:26:24 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 12 December 2010 09:43So this is basically adding more combinations of arguments and parameters to Callback, similar to what was alredy there with callback1, right?

OK, why not. Patch applied.

Mirek

Yes, Mirek.

I can improve this functionality with simple functions (not method from class) too.

I propose to update example too for demonstrate how it can be used.

Subject: Re: Callback (THISBACK) Improve
Posted by mirek on Sun, 12 Dec 2010 14:55:09 GMT
View Forum Message <> Reply to Message

tojocky wrote on Sun, 12 December 2010 07:26

I propose to update example too for demonstrate how it can be used.

OK, update it

Mirek

Subject: Re: Callback (THISBACK) Improve
Posted by kohait00 on Mon, 13 Dec 2010 16:13:13 GMT
View Forum Message <> Reply to Message

i was waiting for this feature long time, meanwhile, i had to pass all the other combinations grouping them together in a TupleX<>.
just to save me the hassle with those Calback modifications..

this makes things quite easy now, thanks..

Subject: Re: Callback (THISBACK) Improve
Posted by kohait00 on Fri, 17 Dec 2010 10:01:21 GMT
View Forum Message <> Reply to Message

i found out that the last change didnt really work and started to scratch Callback layer and found out many inconsistancies in name and usage.. so the past day i was restructuring the code blocks and renaming what was needed.

also, the changes recently made did not support the in-callback type conversion, i.e specifying a callback for float but parametrizing it with int.. this is corrected.

further more, i have extended Callback and Gate and unified their definitions.

since the code grew quite a bit with all that, i have grouped the stuff by count of arguments passing to THISBACK. so you will find a Callback0.h for THISBACK stuff, Callback1.h for THISBACK1, etc..
Callback.h simply includes them all..

just see the code. i think it's pretty clean now and easy to extend if needed, since all Callbacks now have same respective template type names accordingly. i havent changed the code in terms of function, but some naming issues to be more clean, i.e.

struct CallbackMethodActionArg1_2 : public Callback2Action<P1, P2> {}

//to

struct Callback2MethodActionArg1 : public Callback2Action<P1, P2> {}

because they made name clashes when extending..

for testing, i have compiled a major application, and also TheIDE, which both still work and also created a test app where all the combinations are listed to check right compilation. there, you will find the new added combinations and those still not supported.

i also added a STDBACK, STDBACK1, etc. helper, when dealing with non THISBACK functions (because i always have to remeber the syntax for those..)

EDIT:

there are now versions to reduce a callback's attributes by one, specifying a parameter from right, yiedling a callback with one parameter less..

i also found some templates for non thiscall's that use class R template type for return value, whereas callbacks are supposed to be void returning. is this a remanent of old times or is it 'by design'?

File Attachments

1) Callback.rar, downloaded 360 times

Subject: Re: Callback (THISBACK) Improve
Posted by kohait00 on Mon, 20 Dec 2010 09:59:59 GMT
View Forum Message <> Reply to Message

noticed that some reducing argument templates were still missing.
here comes the additional in Callback1.h, simply replace file..

this makes things like this possible

```
Callback cb0;
Callback1<double> cb1;
Callback2<double, double> cb2;
Callback3<double, double, double> cb3;
Callback4<double, double, double, double> cb4;

Gate gt0;
Gate1<double> gt1;
Gate2<double, double> gt2;
Gate3<double, double, double> gt3;
Gate4<double, double, double, double> gt4;

//args reduction

//1 arg to 0 args
cb0 = callback1(cb1, 1);

//2 args to 1 arg
cb1 = callback1(cb2, 2);
cb0 = callback2(cb2, 1, 2);

//3 args to 2 args
cb2 = callback1(cb3, 3); //new
cb0 = callback3(cb3, 1, 2, 3); //new

//4 args to 3 args
cb3 = callback1(cb4, 4); //new
cb0 = callback4(cb4, 1, 2, 3, 4); //new



//1 arg to 0 args
gt0 = callback1(gt1, 1); //new

//2 args to 1 arg
gt1 = callback1(gt2, 2); //new
gt0 = callback2(gt2, 1, 2); //new

//3 args to 2 args
gt2 = callback1(gt3, 3); //new
gt0 = callback3(gt3, 1, 2, 3); //new

//4 args to 3 args
gt3 = callback1(gt4, 4); //new
gt0 = callback4(gt4, 1, 2, 3, 4); //new
```

File Attachments
1) Callback1.h, downloaded 377 times

Subject: Re: Callback (THISBACK) Improve
Posted by tojocky on Mon, 20 Dec 2010 16:06:13 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Fri, 17 December 2010 12:01i found out that the last change didnt really work and started to scratch Callback layer and found out many inconsistancies in name and usage.. so the past day i was restructuring the code blocks and renaming what was needed.

also, the changes recently made did not support the in-callback type conversion, i.e specifying a callback for float but parametrizing it with int.. this is corrected.

further more, i have extended Callback and Gate and unified their definitions.

since the code grew quite a bit with all that, i have grouped the stuff by count of arguments passing to THISBACK. so you will find a Callback0.h for THISBACK stuff, Callback1.h for THISBACK1, etc..
Callback.h simply includes them all..

just see the code. i think it's pretty clean now and easy to extend if needed, since all Callbacks now have same respective template type names accordingly. i havent changed the code in terms of function, but some naming issues to be more clean, i.e.

struct CallbackMethodActionArg1_2 : public Callback2Action<P1, P2> {}

//to

struct Callback2MethodActionArg1 : public Callback2Action<P1, P2> {}

because they made name clashes when extending..

for testing, i have compiled a major application, and also TheIDE, which both still work and also created a test app where all the combinations are listed to check right compilation. there, you will find the new added combinations and those still not supported.

i also added a STDBACK, STDBACK1, etc. helper, when dealing with non THISBACK functions (because i always have to remeber the syntax for those..)

EDIT:

there are now versions to reduce a callback's attributes by one, specifying a parameter from right,

yiedling a callback with one parameter less..

i also found some templates for non thiscall's that use class R template type for return value, whereas callbacks are supposed to be void returning. is this a remanent of old times or is it 'by design'?

Nice work!

Subject: Re: Callback (THISBACK) Improve
Posted by kohait00 on Tue, 21 Dec 2010 14:45:41 GMT
View Forum Message <> Reply to Message

i'm still wondering about those templated return type callbacks that exist..do they have any practical usage besides making arbitrary methods (at least up to Callback1) usable for Callback api? i mean, having a int MyClass::DoSth() is not able to be set as a callback target otherwise..

does anyone have some experience with it?

it'd be great to specify this where needed, maybe sth like

Callback cb = THISBACK(IgnoreReturn(MyMethod));

Subject: Re: Callback (THISBACK) Improve
Posted by tojocky on Wed, 22 Dec 2010 20:56:27 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 21 December 2010 16:45i'm still wondering about those templated return type callbacks that exist..do they have any practical usage besides making arbitrary methods (at least up to Callback1) usable for Callback api? i mean, having a int MyClass::DoSth() is not able to be set as a callback target otherwise..

does anyone have some experience with it?

it'd be great to specify this where needed, maybe sth like

Callback cb = THISBACK(IgnoreReturn(MyMethod));

It is possible. I can give an example and a little change in callback to make it easy to use.

Subject: Re: Callback (THISBACK) Improve

Posted by mirek on Sat, 25 Dec 2010 10:01:58 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Mon, 13 December 2010 11:13i was waiting for this feature long time, meanwhile, i had to pass all the other combinations grouping them together in a TupleX<>.

Actually, and that is bad in what sense?

I am not about extending stuff here, but for 10 years we could quite happilly live with the current set, occasionally packing the set of parameters into some struct. I would say that if your code overflows with multiparameter callbacks, there is something bad with it...

Subject: Re: Callback (THISBACK) Improve
Posted by mirek on Sat, 25 Dec 2010 10:03:54 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 21 December 2010 09:45i'm still wondering about those templated return type callbacks that exist..do they have any practical usage besides making arbitrary methods (at least up to Callback1) usable for Callback api? i mean, having a int MyClass::DoSth() is not able to be set as a callback target otherwise..

does anyone have some experience with it?

it'd be great to specify this where needed, maybe sth like

Callback cb = THISBACK(IgnoreReturn(MyMethod));

You can always add a second method to ignore parameter. Such one-liner is simple.

Subject: Re: Callback (THISBACK) Improve
Posted by kohait00 on Sun, 26 Dec 2010 09:04:44 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 25 December 2010 11:01kohait00 wrote on Mon, 13 December 2010 11:13i was waiting for this feature long time, meanwhile, i had to pass all the other combinations grouping them together in a TupleX<>.

Actually, and that is bad in what sense?

I am not about extending stuff here, but for 10 years we could quite happilly live with the current set, occasionally packing the set of parameters into some struct. I would say that if your code

overflows with multiparameter callbacks, there is something bad with it...

no it's not bad :)no question about that. it only was bit more of a hassle and write, to put them all together.

my motivation was to clean up actually. the extending brought me there. i often ran into situations where a callbackX comination was needed but not present and i also wanted to extend, but hesitated to start to wrap my mind around callback infrastructure in depths. when tojocky extended the callbacks it was more clearly visible where and what to do. so i have seen that the code blocks could be restructured a bit more to easy find and extend things if needed.

i might take a look and remove/separate all the callbacks extensions in an extra package, sth. like a CallbackEx, for those who like it. but this still would involve to update some of them because of name clashes.

Subject: Re: Callback (THISBACK) Improve
Posted by kohait00 on Tue, 18 Jan 2011 09:24:38 GMT
View Forum Message <> Reply to Message

i think there is even more potential in the Callback structs which is not yet used, because it doesnt offer some combinations (just as tojockey has been extending it)

i.e i am using callbacks as implicit grouping of information instead of setting up arrays of structs that group them.

Quote:
if your code overflows with multiparameter callbacks, there is something bad with it...

this is not always the case

have you taken a look at the code? i think it's more structured now.

i even might think of keeping things separated, means that the additional templates could go in an extra package, CallbackExtend or sth. but for this, there is still the need to have a naming convention, so users can extend theirs on their own.

should i prepare a package for that? are you interested?

Quote:
for 10 years we could quite happilly live with the current set

i'm all with it, i like the upp code really a lot, it's tried and powerfull. yet on some edges it could be more 'user' friendly. i often ran into 'well, this Callback combination is not possible'.
so i lined the things up for 98% usage..i dare to say the current is maybe 85%..

Subject: Re: Callback (THISBACK) Improve
Posted by mirek on Fri, 28 Jan 2011 09:31:42 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 18 January 2011 04:24
should i prepare a package for that? are you interested?

Me personally? Not really. Today I barely catch up with things i NEED, not much time to play with things I theoretically might need in future

But please, while my influence on what has or has not to be in uppsrc is undebatable, I am not the only one to decide. So if anybody else desires more complex callback combinations to be covered, I will get back to this and hapilly put it into Core.

Until then, I recommend Bazaar. I guess that soon we will start something like "one package per month" voting process to get things from Bazaar to uppsrc. Fair enough?

Quote:
so i lined the things up for 98% usage..i dare to say the current is maybe 85%..

Example?

Mirek

Subject: Re: Callback (THISBACK) Improve
Posted by kohait00 on Tue, 01 Feb 2011 21:39:54 GMT
View Forum Message <> Reply to Message

i tried to organize the changes in a visible way..maybe it's more clear now what i mean.. and what i have done..

the problem of extending it in bazaar is the name clash with tojockeys adds. i have changed the name the logical extend.

File Attachments
1) cb.gif, downloaded 940 times

2) gate.GIF, downloaded 891 times
3) callbacks.xls, downloaded 383 times

Subject: Re: Callback (THISBACK) Improve
Posted by tojocky on Wed, 02 Feb 2011 07:34:43 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 01 February 2011 23:39i tried to organize the changes in a visible way..maybe it's more clear now what i mean.. and what i have done..

the problem of extending it in bazaar is the name clash with tojockeys adds. i have changed the name the logical extend.

I Agree with you, But, Mirek is right! In this case we have a problem with overloaded functions. To be more clear, we need to keep old functionality and to provide new functions like:
THISBACKDETR(rettype, method) Callback(new CallbackMethodAction<CLASSNAME, rettype (CLASSNAME::*)()>(this, &CLASSNAME::method)) // can accept functions with return type, but cannot get the returned value
THISBACKDETRP0ARG1(rettype, method, arg1_value)
THISBACKDETRP0ARG2(rettype, method, arg1_value, arg2_value)
THISBACKDETRP0ARG3(rettype, method, arg1_value, arg2_value, arg3_value)
....
THISSIGNDETR(rettype, method) // will return type like Gate but arbitrary type
THISSIGNDETRP0ARG1(rettype, method, arg1_value) // will return type like Gate but arbitrary type

OK, I propose to create a package in bazaar with our realization and a simple demo.

Subject: Re: Callback (THISBACK) Improve
Posted by kohait00 on Wed, 02 Feb 2011 08:34:30 GMT
View Forum Message <> Reply to Message

my problem is that i couldnt figure out why the others callbackactions in params 0 row are not made with template class R return type..

if thats the matter i can change it quite fast to the class R thing again.. does it mess up with the Gate return types?

Subject: Re: Callback (THISBACK) Improve

Posted by kohait00 on Thu, 10 Feb 2011 10:26:53 GMT
View Forum Message <> Reply to Message

tojocky wrote on Wed, 02 February 2011 08:34
we need to keep old functionality and to provide new functions
i totally agree with you..thats why i dont quite understand why the hesitation. i'm using the changes in my projects for some time now. ide compiles and works as before.

* the provided cleanup does not change functionality at all (except for 1 case: class R return types as marked below), it extends stuff to some more use cases.

* if anyone could tell the reason for the class R return type templates from 2-args row and forth (and why 1 arg row is normal void), i could change that to it's previous state, making things *totally* compatible..

it's a cleanup, consider people trying to understand callback. with current sources..that's not easy.

Subject: Re: Callback (THISBACK) Improve
Posted by kohait00 on Thu, 28 Apr 2011 11:53:33 GMT
View Forum Message <> Reply to Message

sorry to poke again..
this one is still pending.

i'm actually using this code for quite a long time now, and it makes no problems. besides the return value changes its the old code reordered. in the new layout of files the finding and extending of own stuff is a lot esier..

the 'class R' return value issue isn't explained so i have to consider it kinda legacy from testing times...which should be fixed for cleaning and has been adressed in this cleanup as well.

so what's the deal no taking it upstream?
(i want to become upstream again, have several changes laying at my side)..

the changes are really few (see what has been added in red for Callback, it's small (the remainder is only the same thing for Gate).

cheers

Subject: Re: Callback (THISBACK) Improve
Posted by mdelfede on Tue, 04 Feb 2014 22:55:41 GMT
View Forum Message <> Reply to Message

Hi.... reviving this one because I'm in need of some additions to Callbacks mechanics. In detail:

- some way to call functions with returned value (and get the value....).

- maybe an argument more...it would be handy sometimes. So Callback5.

Did somebody implement at least the first one ?

Ciao

Max

Subject: Re: Callback (THISBACK) Improve
Posted by zsolt on Wed, 05 Feb 2014 15:50:27 GMT
View Forum Message <> Reply to Message

You can send a Callback(N) object, I think. Or a pointer to your variable to set?
For the second one: have you tried using structs instead of a lot of arguments?

Subject: Re: Callback (THISBACK) Improve
Posted by mdelfede on Wed, 05 Feb 2014 16:28:50 GMT
View Forum Message <> Reply to Message

Yep... I'm using a tuple, now, and an helper function with a return value reference as a parameter.
Quite unelegant....

Subject: Re: Callback (THISBACK) Improve
Posted by piotr5 on Sun, 09 Feb 2014 11:09:08 GMT
View Forum Message <> Reply to Message

I think Callback should be implemented with variable number of parameters in c++11 once u++ is ready for that.

Subject: Re: Callback (THISBACK) Improve
Posted by mdelfede on Sun, 09 Feb 2014 11:15:57 GMT
View Forum Message <> Reply to Message

GCC and MSC have already variadic macros, even if it's not a standard feature. I'm using it in my PolyXML stuff and they work great, but even without them the Callbacks are quite good.
The only thing I miss are handling of functions returning values (don't know if it's possible....) and some more arguments.

Subject: Re: Callback (THISBACK) Improve
Posted by piotr5 on Sun, 09 Feb 2014 14:43:59 GMT
View Forum Message <> Reply to Message

I were thinking of variadic templates in combination with variadic functions, so "more parameters" are not necessary since infinitely many are allowed. as for return value, I am not sure it can still be called callback when it's returning non-boolean value. the caller usually doesn't know who is being called. better use THISBACK for your own class and let the callback store some value inside of that class instead of returning it. but I agree some generic code doing that for global functions would be nice. new Templates4U project on bazaar?