# c/c++ programming by examples

we share the c/c++ coding by examples

## simple callback function

Posted on **October 11, 2007**

Callback function is hard to trace, but sometimes it is very useful. Especially when you are designing libraries. Callback function is like asking your user to gives you a function name, and you will call that function under certain condition.

For example, you write a callback timer. It allows you to specified the duration and what function to call, and the function will be callback accordingly. “Run myfunction() every 10 seconds for 5 times”

Or you can create a function directory, passing a list of function name and ask the library to callback accordingly. “Callback success() if success, callback fail() if failed.”

Lets look at a simple function pointer example

```
01  void cbfunc()
02  {
03      printf("called");
04  }
05
06  int main ()
07  {
08       /* function pointer */
09      void (*callback)(void);
10
11      /* point to your callback function */
12      callback=(void *)cbfunc;
13
14     /* perform callback */
15     callback();
16
17     return 0;
18  }
```

**How to pass argument to callback function?**

Observered that function pointer to implement callback takes in void *, which indicates that it can takes in any type of variable including structure. Therefore you can pass in multiple arguments by structure.

```
01  typedef struct _myst
02  {
03      int a;
04      char b[10];
05  }myst;
06
07  void cbfunc(myst *mt)
08  {
09      fprintf(stdout,"called %d %s.",mt->a,mt->b);
10  }
11
12  int main()
13  {
14
15      /* func pointer */
16      void (*callback)(void *);
17
18      //param
19      myst m;
20      m.a=10;
21      strcpy(m.b,"123");
22
23      /* point to callback function */
24      callback = (void*)cbfunc;
25
26      /* perform callback and pass in the param */
27      callback(&m);
28
29      return 0;
30
31  }
```

This entry was posted in Uncategorized by **mysurface**. Bookmark the **permalink [http://cc.byexamples.com/2007/10/11/simple-callback-function/]** .

31 THOUGHTS ON "SIMPLE CALLBACK FUNCTION"

Vladimir Grekhov
on **November 10, 2007 at 9:26 am** said:

Thank you for the example, please make a change in the first snap shot of code.

void (*callback)(void *);
to
void (*callback)(void);

compumaster - cebu
on **July 22, 2008 at 11:22 am** said:

yup…i agree with vladimir…but we can also do it like this…

```
void cbfunc()
{
printf("called");
}

int main ()
{
/* function pointer */
void (*callback)();

/* point to your callback function */
callback = &cbfunc;

/* perform callback */
callback();

return 0;
}
```

**mysurface**
on **July 22, 2008 at 11:17 pm** said:

Thanks Vladimir, corrected.

@cebu, it seems more clean 

Aditya
on **October 8, 2008 at 10:12 pm** said:

its an awesome description.. thankyou very much all

freahy
on **October 30, 2008 at 8:33 am** said:

It ‘s very useful , thank you everybody.

**Bhimsen**
on **December 9, 2008 at 2:30 pm** said:

good simple example

Ashok kumar reddy
on **December 13, 2008 at 5:15 pm** said:

its Good understanding call back function in simple way

nitin
on **December 22, 2008 at 8:09 pm** said:

if your function is ” cbfunc(myst *mt)”

then ur callback function should be “void (*callback)(myst *) ” NOT this “void (*callback)(void) “

Aditya Kumar Padhi
on **January 22, 2009 at 4:08 pm** said:

i m fine with his code. he explained in a simple manner how to use callback funciton.

Anup
on **March 23, 2009 at 4:47 pm** said:

Hello All
i am getting following error while compiling it on linux
error: invalid conversion from `void*’ to `void (*)(myst*)’
can anybody plz suggest why it is coming

Rajesh
on **April 2, 2009 at 8:17 pm** said:

You’ve kept the whole thing really simple and the explanation is very neatly written.
Keep the good work up.

Sanjeet

on **April 3, 2009 at 2:32 pm** said:

ya i agree with nitin i.e

if your function is " cbfunc(myst *mt)"

then ur callback function should be "void (*callback)(myst *) " NOT this "void (*callback)(void) "

will

on **April 27, 2009 at 11:38 pm** said:

I am new to Callback, can you make sure your code fixed and compiled, then post it again please. I learn from example, however, your code does not compiled. Thanks!

micah

on **June 25, 2009 at 4:40 am** said:

How do you put this in a class? When I try to make the callback function pointer a member variable pointing to a function in another class, it says 'void (MyClass::)()' does not match 'void(*)()'

micah

on **June 25, 2009 at 5:04 am** said:

Oh, I was forgetting the (void*) in the '(void*)blankfunc'. But when I have the (void*) it says "invalid use of member( did you forget the '&'?)

rakesh
on **March 3, 2010 at 5:05 pm** said:

I have a doubt.

i can call the function “cbfunc(myst *mt)” directly.
then why i am assigning “cbfunc(myst *mt)” function to “callback” function pointer & then calling it.

rakesh
on **March 3, 2010 at 5:08 pm** said:

its not the example of a call back function.it is an example of function pointer

Prem
on **April 23, 2010 at 10:27 am** said:

Actually FP is the way to implement Callback. So the way to implement callback is the way in which you call the pointer variable (with/without argument) which is the refference to the function on which it is assigned..

Kumar
on **September 3, 2010 at 1:24 pm** said:

This is how in embedded systems call back functions are used. may seems to be little bit complex but it is easy if u read the comments first

```
typedef struct
{
int a;
int b;
int (* callback)(int a,int b);
}struct_b;

/*Assume that this is ur library i,e lower level function*/
int mylib(struct_b );

/*Assume that this ur higher level function*/
int add(int a,int b);

void main(void)
{
int a,b;
struct_b st_b;

/*Here u r passing higher level function to lower level function through structure*/
st_b.callback = add;
st_b.a = 10;
st_b.b = 12;
printf("sum of a and b is = %d",mylib(st_b));

}

/*This is ur higher level function definition*/
int add(int a,int b)
{
return a+b;
}

int mylib(struct_b st_b)
{
/*Here u r calling higher level function with the help of call back function*/
return st_b.callback(st_b.a,st_b.b);
}
```

PS
on **January 12, 2011 at 5:21 pm** said:

This example just shows how function pointer is used to call a function. Actually 2 seperate threads or process are required to show callback functions.
One example could be device driver code registering ISR(interupt service routine) with OS and OS calls the ISR(the function) when and interrupt is signalled.

Sudarshan
on **March 17, 2011 at 6:16 am** said:

Please provide a similar example in c++, for example using callback functions for handling interrupts.

Mohan
on **July 12, 2011 at 11:43 am** said:

pretty good explanation in a simple way..

pratima sharma
on **October 2, 2011 at 5:11 pm** said:

nice example

pratima sharma
on **October 2, 2011 at 5:13 pm** said:

Declaring
Declare a function pointer as though you were declaring a function, except with a name like *foo instead of just foo:

```
void (*foo)(int);
```

Initializing
You can get the address of a function simply by naming it:

```
void foo();
func_pointer = foo;
```

or by prefixing the name of the function with an ampersand:

```
void foo();
func_pointer = &foo;
```

Invoking
Invoke the function pointed to just as if you were calling a function.

```
func_pointer( arg1, arg2 );
```

or you may optionally dereference the function pointer before calling the function it points to:

```
(*func_pointer)( arg1, arg2 );
```

Benefits of Function Pointers

Function pointers provide a way of passing around instructions for how to do something
You can write flexible functions and libraries that allow the programmer to choose behavior by passing function pointers as arguments
This flexibility can also be achieved by using classes with virtual functions

kuppesh
on **December 30, 2011 at 6:04 am** said:

We have given very simple and good examples.
Thank you.

kuppesh
on **December 30, 2011 at 6:05 am** said:

You have given good examples

jad halime
on **January 23, 2012 at 3:03 pm** said:

thank you all for the great explanation… its the best on the onternet.. love you all

Anonymous
on **March 16, 2012 at 5:01 am** said:

Yes the example on top explains about FP and not callbacks

DerekTan
on **July 26, 2012 at 4:52 am** said:

perfect tutorial.

Sreeraj
on **July 29, 2012 at 7:29 am** said:

Nice examples. Quite understandable.

virgo
on **August 21, 2012 at 4:10 am** said:

```
#include
#include

//Predeclarations
class cMyProject;

template
class TCallback;

class cCallback
{
public:
virtual void Execute(int Param) const =0;
};

template
class TCallback : public cCallback
{
public:
TCallback() // constructor
```

```
{
pFunction = 0;
}

typedef void (cInstance::*tFunction)(int Param);

virtual void Execute(int Param) const
{
if (pFunction) (cInst->*pFunction)(Param);
else printf("ERROR : the callback function has not been defined !!!!");
}

void SetCallback (cInstance *cInstancePointer,
tFunction pFunctionPointer)
{
cInst = cInstancePointer;
pFunction = pFunctionPointer;
}

private:
cInstance *cInst;
tFunction pFunction;
};

// Some instances of the Callback class
TCallback i_Callback_1;
TCallback i_Callback_2;

class cMyProject{

private:

public:
// the functions of your project
void CallbackFox (int Param);
void CallbackRabbit(int Param);
void TestTheCallback(cCallback *pCallbackFunction, int Param);
void CallbackDemo();

};

void cMyProject::CallbackRabbit(int Param)
{
```

```
char Buf[50];
sprintf(Buf, “Now I’m in Rabbit with Param %d !\n”, Param);
printf(“%s”,Buf);
}

void cMyProject::CallbackFox(int Param)
{
char Buf[50];
sprintf(Buf, “Now I’m in Fox with Param %d !\n”, Param);
printf(“%s”,Buf);
}

void cMyProject::TestTheCallback(cCallback *pCallbackFunction, int Param)
{
pCallbackFunction->Execute(Param * Param);
}

void cMyProject::CallbackDemo()
{

// defining where the callback should jump to
i_Callback_1.SetCallback(this, &cMyProject::CallbackRabbit);
i_Callback_2.SetCallback(this, &cMyProject::CallbackFox);

// now you can pass i_Callback like a pointer to a function
TestTheCallback(&i_Callback_1, 4);
TestTheCallback(&i_Callback_2, 5);
}

int main(int argc, char* argv[])
{

cMyProject* test_ = new cMyProject;
test_ -> CallbackDemo();
delete test_;

return 0;
}
```