Introduction to signal0x

Wang Feng

May 10, 2011

1 Tutorial

1.1 Motivation

This is another wheel invented as an implementation of Observer Pattern written in C++.

This is also a demonstration of the upcoming c++0x.

Features:

1. Thread–safe
2. Faster
3. Non–intrusive connection tracking
4. Automatic disconnection

1.2 Compatibility Note

Signal0x library compiles currently only compatibile with g++4.6 and g++4.7, as it employs many c++0x features such as:

1. variadic template
2. range–based for loop
3. rvalue reference
4. lambda
5. mutex
6. ...

The header file of signal0x is

```
#include <signal.hpp>
```

The namespace of signal0x is

```
using signal0x::signal;
```

A typical commandline compilation is

g++ -o test test.cc -std=c++0x -O2

1.3 Connecting to Signals

1.3.1 First Example – Hello World

```
1  // an ordinary function
2  void hello_world()
3  {
4      std::cout << "Hello World.\n";
5  }
6
7  //(1) a signal with no arguments and a void return value
8  signal0x::signal<void> sig;
9
10 //(2) connect function to signal
11 sig.connect( hello_world );
12
13 //(3) signal triggered
14 sig();
```

This example writes "Hello World." using signals and slots in the following way:

1. create a signal sig, a signal that takes no arguments and has a void return value
2. connect a slot, the hello_world function, to the signal using the connect method
3. the signal is triggered, which in turns invokes hello_world() to print "Hello World.".

1.3.2 Slots

Several kinds of slots can be connected to a signal

1. a function

```
1 #include <signal.hpp>
2 using signal0x::signal;
3
4 void f(){}
5 signal<void> sig;
6 sig.connect( f );
7 sig();
```

2. a functor

```cpp
#include <signal.hpp>
using signal0x::signal;

struct f{ void operator()() const {}};
f f_;
signal<void> sig;
sig.connect( f() );
sig.connect( f_ );
sig();
```

3. a lambda object

```cpp
#include <signal.hpp>
using signal0x::signal;

signal<void> sig;
sig.connect( []{} );//an anonymous lambda object
sig();
```

4. a pointer to a member function

```cpp
#include <signal.hpp>
#include <functional>//for std::bind
using signal0x::signal;

struct f
{
    void ff() {}
};
f f_;
signal<void> sig;
sig.connect( std::bind( &f::ff, &f_) );
sig();
```

1.3.3 Multiple Slots

Multiple slots can be connected to a same signal, and if the signal is triggered, all slots connected to this signal will be called.

Another "hello world" example

```cpp
#include <signal.hpp>
#include <iostream>

void hello{ std::cout << "Hello "; }
void world{ std::cout << "World!"; }

signal0x::signal<void> sig;
```

```
 8 sig.connect( hello );
 9 sig.connect( world );
10 sig(); //print "Hello World!"
```

1.3.4 Ordering Multiple Slots

1.4 Examples

2 Comparison With Boost::signals and sigc::signal

2.1 Grammar Comparison

2.2 Benchmark

3 Design and Implementation