Boost.Signals2

Основные понятия.

- *Сигнал* — это тип данных, который может хранить в себе несколько функций обратного вызова, и вызывать их.
  Слот — это, соответственно, и есть подсоединяемые к сигналу функции.

Для чего нужен Boost.Signals2?

- Сигналы могут быть полезны при разработке приложений с графическим интерфейсом.
- Для обработки Событий.

Пример.

- Этапы.
- **Создаем сигнал sig**
- boost::signals2::signal<void ()> sig;
- **подключаем слот**
- Button mainButton;
- sig.connect(mainButton);
- **Вызываем слоты:**
- sig();

Получаем:

```
#include <iostream>
#include <boost/signals2/signal.hpp>
struct Button
{
    void operator()() const
    {
        std::cout<<"Slot called"<<std::endl;
    }
};
int main()
{
    boost::signals2::signal<void ()> sig;
    Button mainButton;
    sig.connect(mainButton); //Подключаем слот
    sig ();
    return 0;
}
```

Вызов нескольких слотов.

ПОРЯДОК ВЫЗОВА СИГНАЛОВ.

можно разделить слот на два:

```
struct ButtonSlot
{
    void operator()() const
    {
        std::cout<<"Slot ";
    }
};
struct ButtonCall
{
    void operator()() const
    {
        std::cout<<"called"<<std::endl;
    }
};
```

- И вызывать их по очереди
- ButtonFunc mainButtonSlot;
- ButtonCall mainButtonCall;
- sig.connect(mainButtonSlot);
- sig.connect(mainButtonCall);

Порядок вызова.

- The Boost.Signals2 позволяет разместить слоты в группы.
- Для этого мы создаем дополнительный параметр (типа int), который и упорядочивает группы.

- sig.connect(1,ButtonCall());
- sig.connect(0,ButtonSlot());

- Непронумерованные слоты могут быть помещены в начало или конец списка слотов, если использовать boost::signals2::at_front или boost::signals2::at_back соответственно, в качестве последнего параметра в сигнале.
- т.е. sig.connect(Button(),boost::signals2::at_front);
- если и этого не укажем, то уйдет в конец списка.

Сигналы с параметрами и сигналы, возвращающие значения.

Сигналы с параметрами

- Сигналы могут содержать параметры.
- Создадим сигнал, который отправляет два аргумента float в его слоты:
- boost::signals2::signal<void (float, float)> sig;

- затем создадим пару слотов, которые печатают результаты нескольких арифметических действий над этими значениями:
- void print_args(float x, float y)
- { std::cout << "The arguments are " << x << " and " << y << std::endl;
- }
- void print_sum(float x, float y)
- { std::cout << "The sum is " << x + y << std::endl;
- }
- void print_product(float x, float y)
- { std::cout << "The product is " << x * y << std::endl;
- }
- void print_difference(float x, float y)
- { std::cout << "The difference is " << x - y << std::endl;
- }

- и вызываем слоты:
- sig.connect(&print_args);
- sig.connect(&print_sum);
- sig.connect(&print_product);
- sig.connect(&print_difference);
- sig.connect(&print_quotient);
-
- sig(5., 3.);

- Вот, что выведет программа на экран:
- The arguments are 5 and 3
- The sum is 8
- The product is 15
- The difference is 2
- The quotient is 1.66667

Сигналы, возвращающие значения.

- Немного исправим предыдущую программу, чтобы слот возвращали значения арифметических операций:
- float product(float x, float y) { return x * y; }
- float quotient(float x, float y) { return x / y; }
- float sum(float x, float y) { return x + y; }
- float difference(float x, float y) { return x - y; }

- Тогда их вызов будет выглядеть так:
- sig.connect(&product);
- sig.connect(&quotient);
- sig.connect(&sum);
- sig.connect(&difference);
- std::cout << *sig(5, 3) << std::endl;

- Обратите внимание: на экран выведется «2», так как возвращается результат последнего вызова слота.

- Чтобы вывести все результаты
- 1. template<typename Container>
- struct aggregate_values
- {
- typedef Container result_type;
- template<typename InputIterator>
- Container operator()(InputIterator first, InputIterator last) const
- {
- Container values;
- while(first != last) {
- values.push_back(*first);
- ++first; }
- return values;
- }
- };

- 2. boost::signals2::signal<float (float, float),
- aggregate_values<std::vector<float> > > sig;
- sig.connect(&quotient);
- sig.connect(&product);
- sig.connect(&sum);
- sig.connect(&difference);
- std::vector<float> results = sig(5, 3);
- std::cout << "aggregate values: ";
- std::copy(results.begin(), results.end(),
- std::ostream_iterator<float>(std::cout, " "));
- std::cout << "\n";

Управление соединением.

CONNECTION MANAGEMENT.

Отключение сигналов

- Для управления соединением используется класс boost::signals2::connection
- Функция (метод) connect() соединяет сигнал и слот
- Функция disconnect() отсоединяет сигнал и слот

Пример подключения класса

- Вернемся к самому первому примеру и изменим вызов слотов.
- boost::signals2::connection c = sig.connect(HelloWorld());
- std::cout << "c is connected\n";
- sig();
- c.disconnect(); // Disconnect the Button object
- std::cout << "c is disconnected\n";
- sig();

Вот, что выведет программа на экран.

- c is connected
- Slot called
- C is disconnected

Blocking Slots.

- boost::signals2::shared_connection_block временно заблокирует слот.
- Пример:
- #include <boost/signals2/shared_connection_block.hpp>
- …
- { boost::signals2::shared_connection_block block(c);
- std::cout << "c is blocked.\n";
- sig(); }
- std::cout << "c is not blocked.\n";
- sig();

- Замечание:
- std::function также может быть использована для обработки событий.
- Разница между std::function и Boost.Signals2: Boost.Signals2 может связать больше одного слота (обработчика события) с одним сигналом.

Заключение

- 1.Сигналы и слоты очень удобны в том случае, когда нужно уменьшить связность* различных объектов.
- 2. Boost.Signals2 лучший вариант при обработке событий.

- * Свя́зность (англ. *cohesion*) — способ и степень, в которой задачи, выполняемые некоторым программным модулем, связаны друг с другом; мера силы взаимосвязанности элементов внутри модуля.