[C++11]std::packaged_task介绍及使用

原创 godmaycry 2017-06-05 17:16:34 7792 收藏 9 版权

分类专栏: C++

一、std::packaged_task简介

在上一篇，我们介绍了std::promise的使用方法，其实std::packaged_task和std::promise非常相似，简单来说std::packaged_task<F>是对std::promise<T= std::function<F>>中T= std::function<F>这一可调对象(如函数、lambda表达式等)进行了包装，简化了使用方法。并将这一可调对象的**返回结果**传递给关联的std::future对象。
比如上一篇我们用std::promise这样写:

```
//声明一个可调对象T
using T = std::function<int(int)>;        //等同于typedef std::function<int(int)> T;

//函数
int Test_Fun(int iVal)
{
    std::cout << "Value is:" << iVal << std::endl;
    return iVal + 232;
}

//声明一个std::promise对象pr1，其保存的值类型为int
std::promise<T> pr1;
//声明一个std::future对象fu1，并通过std::promise的get_future()函数与pr1绑定
std::future<T> fu1 = pr1.get_future();
```

那么使用std::packaged_task就简化了很多:

```
//函数
int Test_Fun(int iVal)
{
    std::cout << "Value is:" << iVal << std::endl;
    return iVal + 232;
}

//声明一个std::packaged_task对象pt1，包装函数Test_Fun
std::packaged_task<int(int)> pt1(Test_Fun);
//声明一个std::future对象，包装Test_Fun的返回结果，并与pt1关联
std::future<int> fu1 = pt1.get_future();
```

注意：使用std::packaged_task关联的std::future对象保存的数据类型是**可调对象的返回结果类型**，如示例函数的返回结果类型是int，那么**声明为std::future<int>**，而不是std::future<int(int)>。

二、代码示例

```
#include <iostream>
#include <future>
```

```
#include <chrono>
#include <functional>

int Test_Fun(int a, int b, int &c)
{
        //a=1,b=2,c=0

        //突出效果，休眠5s
        std::this_thread::sleep_for(std::chrono::seconds(5));

        //c=233
        c = a + b + 230;

        return c;
}

int main()
{
        //声明一个std::packaged_task对象pt1，包装函数Test_Fun
        std::packaged_task<int(int, int, int&)> pt1(Test_Fun);
        //声明一个std::future对象fu1，包装Test_Fun的返回结果类型，并与pt1关联
        std::future<int> fu1 = pt1.get_future();

        //声明一个变量
        int c = 0;

        //创建一个线程t1，将pt1及对应的参数放到线程里面执行
        std::thread t1(std::move(pt1), 1, 2, std::ref(c));

        //阻塞至线程t1结束(函数Test_Fun返回结果)
        int iResult = fu1.get();

        std::cout << "执行结果：" << iResult << std::endl;       //执行结果：233
        std::cout << "c：" << c << std::endl;   //c：233

        return 1;
}
```

可以看出，当保存的数据类型是可调对象时，使用std::packaged_task比std::promise更简洁。