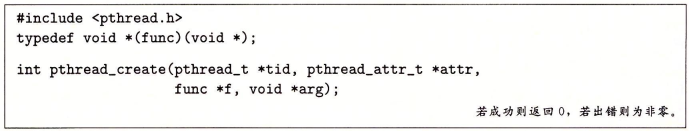
除了函数指针之外，C++还提供了一种实现回调函数的方法，那就是函数对象。函数对象，是指重载了()操作符的普通类对象。

**把要执行的任务写成一个函数，将这个函数和某一时间或中断建立关联**。这个函数就是回调函数。



pthread\_create中的f就是回调函数。还有sort的compare函数等等。

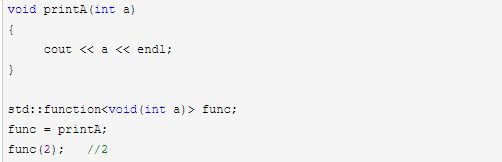
<https://blog.csdn.net/xiaoyink/article/details/79348806>

std::function是一种通用，多态的函数封装类模板，实现了一个泛型的回调机制。std::function的实例可以对任何可以调用的目标实体进行存储，赋值和调用操作。**这些目标实体包括普通函数、Lambda表达式、函数指针、以及其他函数对象。**std::function对象是对C++中现有的可调用实体的一种类型安全的包裹（像函数指针这类可调用实体，是类型不安全的）。

通常std::function是一个函数对象类，它包装其它任意的函数对象，被包装的函数对象具有类型为T1, …,TN的N个参数，并且返回一个可转换到R类型的值。std::function使用 模板转换构造函数接收被包装的函数对象；特别是，闭包类型可以隐式地转换为std::function。

function<int(int,int)>表示声明了一个function，他可以表示接受两个int参数，返回一个int类型的可调用对象。

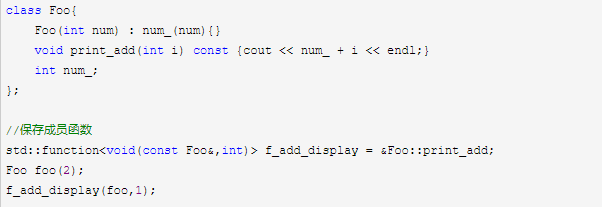
1、保存普通函数



2、保存lamada表达式



3、保存成员函数



常常使用auto代替std::function<void(const Foo&,int)>

4、bind，可以将bind视为一个通用的函数适配器，他接受一个可调用对象，生成一个新的可调用对象来适应原对象的参数列表。

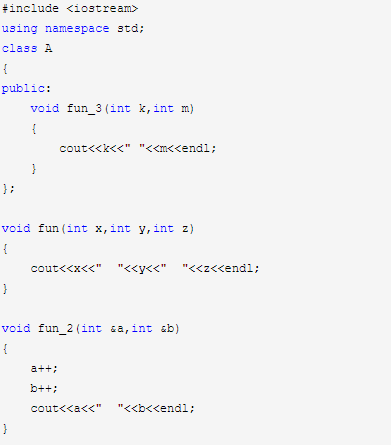
一般形式：

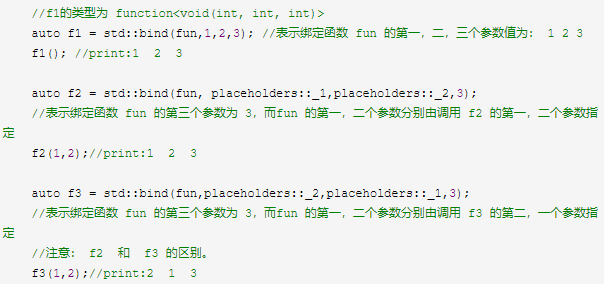
auto newCallable = bind(callable,arg\_list);

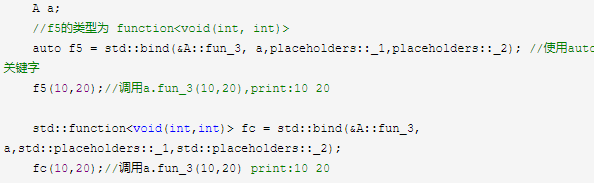
其中，newCallable本身是一个可调用对象，arg\_list是一个逗号分隔的参数列表，对应给定的callable的参数。

即，当我们调用newCallable时，newCallable会调用callable,并传给它arg\_list中的参数。

arg\_list可能包含形如\_n的名字，称为占位符，表示参数的位置，\_1为newCallable的第一个参数，\_2为newCallable的第二个参数。占位符只是表示有n个参数，但是现在不指定，等到调用的时候才传入。



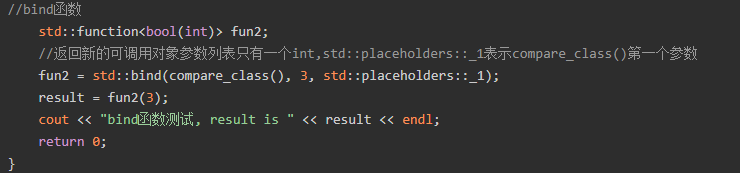


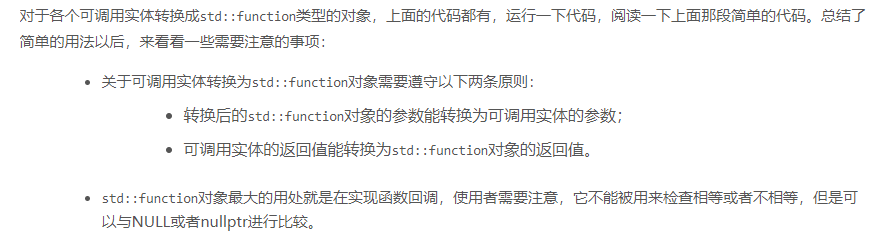


也就是说，对于类成员函数和普通函数来说，可以直接将其传入function对象进行调用，也可以通过bind生成一个可调用对象传入function对象。

**对于成员函数来说，第一个参数是成员函数的调用对象。为了方便使用，常使用bind函数，并且需要实例化对象，成员函数加取地址符。**

**仿函数bind**





threadpool03相比于threadpool98的变化

1、用function和bind实现回调函数而不是函数指针

2、将每个任务封装成一个函数对象，函数对象实现run方法。

3、有一个疑问：

声明function<void()>

但是将bind(int,char\*)赋值给它。

function<void()>指向任意参数？？目前来看是的，但没有找到出处

int run(int i,const char\*p){  
 cout<<"exec "<< i<<endl;  
}  
  
int run1(int a,int b,const char\*p){  
 cout<<"exec "<< a+b<<endl;  
}

function<void()> f = bind(&run,10,"hello");  
f();  
function<void()> f1 = bind(&run1,10,10,"helloworld");  
f1();