1. 右值引用。

左值：指表达式结束后依然存在的**持久对象**，**可以取地址**，具名变量或对象 。

右值：表达式结束后就不再存在的**临时对象**，**不可以取地址**，没有名字。

左值引用：type &引用名 = 左值表达式；

右值引用：type &&引用名 = 右值表达式；

Eg:

 // 右值引用-右值

  int&& rr = 42;

int i = 10;

   int&& rr1 = std::move(i);

    //const左值引用--右值

   const int& c = 42;

   //左值引用--左值

   int& lr = rr;

    //注意：右值引用也是变量，所以rr是左值。

Move1.cpp

1. std::move()。

【目的】使代码运行更高效。

【功能】**将一个左值引用强制转化为右值引用，继而可以通过右值引用使用该值，以用于移动语义**。

【优点】std::move语句可以将左值变为右值而**避免拷贝构造**。

std::move是**将对象的状态或者所有权从一个对象转移到另一个对象**，

只是转移，**没有内存的搬迁或者内存拷贝。**

<http://blog.gssxgss.me/cpp11-move-constructor-and-std-move-1/>

1. std::result\_of()

【功能】

用于在编译的时候推导出一个可调用对象的返回值类型.主要用于模板编写中。

result\_of1.cpp

1. std::future std::packaged\_task() std::promise

future1.cpp future2.cpp future.png

手机屏幕截图

描述已自动生成

std::packaged\_task<int(int)>

1. std::bind()

将原来的可调用对象和其参数进行绑定，形成一种新的调用形式

（1）改变参数个数。

（2）改变参数调用顺序。

bind1.cpp bind2.cpp

auto task = std::make\_shared<std::packaged\_task<return\_type>>(

std::bind(std::forward<F>(f), std::forward<Args>(args)...));

ThreadPool

1. 执行可调用对象的可变参数个数和类型。
2. 执行可调用对象的可变返回值类型。
3. 获取每个thread worker的返回值。