c++引用折叠

原创 kupeThinkPoem 于 2021-08-27 09:03:49 发布 5520 收藏 30

分类专栏： 智能指针源码解析 文章标签： c++

智能指针源码解析 专栏收录该内容 16 订阅 39 篇文章

目录

一、引用折叠

由于存在T&&这种万能引用类型，当它作为参数时，有可能被一个左值引用或右值引用的参数初始化，这是经过类型推导的T&&类型，相比右值引用(&&)会发生类型的变化，这种变化就称为引用折叠。

1.所有右值引用折叠到右值引用上仍然是一个右值引用。（A&& && 变成 A&&）
2.所有的其他引用类型之间的折叠都将变成左值引用。（A& & 变成 A&; A& && 变成 A&; A&& & 变成 A&）

二、示例解析

要说引用折叠，首先得说右值引用(在看这个之前需要了解C++11中左值，右值的概念)。它是C++11出现的新概念，声明类型的方法是：T&&，具体信息可以看下面的代码：

```
Class A
{
    A()
    {// do something}
};

A GetA()
{
    return A();
}

int main()
{
    A a1 = GetA();   // a1是左值
    A&& a2 = GetA(); // a2是右值引用
    return 0;
}
```

a1是左值，在构造时使用了GetA() 产生的临时对象，之后GetA()产生的临时对象会销毁。

a2是右值引用，其指向的就是GetA()所产生的对象，这个对象的声明周期是和a2的声明周期是一致的。即少了临时对象，从而省去了临时对象的构造和析构。

由此可见右值引用的好处，在新代码中，右值引用是值得大力使用的。但是，在使用的时候，有例外情况了:T&&并不是一定表示右值，比如，如果它绑定的类型是未知的话，既可能是左值，又可能是右值。比如：

```
template<typename T>
void f(T&& param)
{
  std::cout<<param;
}

f(10); // 10是右值   int &&
int x = 10;
```

```
9 f(x);  // x是左值    int &
```

以上这种万能引用类型(param的类型)能万能引用不同类型的参数,这种类型必须被初始化,而它是左值还是右值则取决于它的初始化，如果被左值初始化，那么它就是左值，反之亦然。那么什么时候是左值，什么时候是右值，就需要进行类型推导才知道。

有的人会问，我传入的是一个左值a,并不是一个左值引用，为什么 编译器 会推导出T 为int &呢。首先，模板函数参数为 T&& param,也就是说，不管T是什么类型，T&&的最终结果必然是一个引用类型。如果T是int, 那么T&& 就是 int &&；如果T为 int &，那么 T &&(int& &&) 就是&，如果T为&&,那么T &&(&& &&) 就是&&。很明显，接受左值的话，T只能推导为int &。

总结一下，万能引用就是利用模板推导和引用折叠的相关规则，生成不同的实例化模板来接收传进来的参数。

三、参考:

https://www.zhihu.com/question/40346748/answer/88672920

引用折叠和完美转发 - 知乎