|  |  |
| --- | --- |
| 1. 理解std::move 2. Std::move是如何定义的 3. Std::move是如何工作的 4. 从一个左值static\_cast到一个右值引用是允许的 | Template <typename T>  Typename remove\_reference<T>::type&& move(T&& t)  {  Return  static\_cast<typename remove\_reference<T>::type&&>(t);  }  String s1(“sdfsd”),s2;  S2 = std::move(string(“bye!”));  S2 = std::move(s1); |
| 1. 理解std::move 2. 标准库move函数是使用右值引用模板的例子 3. 不能直接将一个右值引用绑定到一个左值引用上，但可以将一个左值引用绑定到右值引用上，move的就是T&&，并且是函数模板 4. Std::move是如何定义的 5. 通过引用折叠可以与任何类型的实参匹配 6. Std::move是如何工作的 7. 第一个赋值中,传递的是右值 8. T为string 9. 返回类型为string&& 10. 参数类型是T&&   String&& move(string &&t)  static\_cast<typename remove\_reference<T>::type&&>(t);  static\_cast<string&&)(t);t是T&&，这转换没有什么问题。   1. 第二个赋值中，传递的是左值 2. T为string& 3. 返回类型为string&& 4. 参数类型string & &&折叠为string&   String&& move(string &t)  这就是一开始的问题：不能直接将一个右值引用绑定到一个左值引用上。  所以有：static\_cast<typename remove\_reference<T>::type&&>(t);  Static\_cast<string&&>(t);,t的类型是string&  Cast将其转换为string&&   1. 从一个左值static\_cast到一个右值引用是允许的 2. Static\_cast，可以针对右值引用，显式将一个左值转换为一个右值引用 3. 将一个右值引用绑定到一个左值的特性允许他们截断左值：就是让左值引用脱去变为右值引用接管。是安全的 | |