|  |  |  |  |
| --- | --- | --- | --- |
|  | **Implementation** | **lvalue** | **rvalue** |
| **Method 1**  **Only value** | Foo Fun(Foo foo){  //change foo  Return foo;  } | Foo foo = Fun(foo1);  //one copy in parameter  //one move in return | Foo foo = Fun(Foo{});  //one ctor in parameter with copy-elision  //one move in return. |
| **Method 2**  **overlaod** | Foo Fun(Foo& foo){  Foo foo1(foo);  //change foo1;  Return foo1;  } | Foo foo = Fun(foo1);  //one copy in foo1;  //copy elision in return | XXX |
| Foo Fun(Foo&& foo){  Return std::move(foo);  Return foo;  } | XXX | Foo foo = Fun(Foo{});  // with std::move, one move  // without move, one copy |
| **Method3**  **const reference** | Foo Fun(const Foo& foo){  Foo foo1(foo);  //change foo1;  Return foo1;  } | Foo foo = Fun(foo1);  //one copy in foo1  //copy elision in return | Foo foo = Fun(Foo{});  //one copy in foo1  //copy elision in return |

|  |  |  |  |
| --- | --- | --- | --- |
| Left value input | ctor | Copy | move |
| Method1 | 1 | 1 | 1 |
| Method2 | 1 | 1 | 0 |
| Mehtod3 | 1 | 1 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| Right value input | ctor | Copy | move |
| Method1 | 1 | 0 | 2(1) |
| Method2 | 1 | 0 | 1 |
| Method3 | 1 | 1 | 0 |

* For method 1 for rvalue, if you use -fno-elide-constructors, then it use 2 move, otherwise use 1 move
* Method 2 is the best, but you need to write two overload function.
* If don’t use method 2. Previous lessons told us that reference is more efficient than value(it can avoid coping). But in our specific scenario(we still copy inside the function even we use reference), if move is cheaper than copy, then method3 is better. Although for lvalue, It use one more move, but for rvalue, it also use move, not copy.