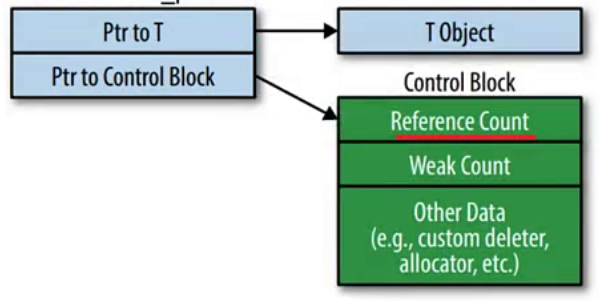
1. **智能指针**



shared\_ptr内部包含两个指针，一个指向对象，另一个指向控制块(control block)，控制块中包含一个引用计数(reference count),一个弱计数(weak count)和其它一些数据。Ref count访问数据是线程安全的，但是修改数据不是线程安全的



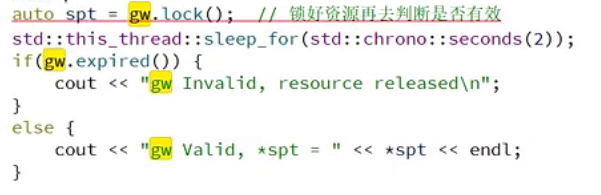
S.get()返回的是对象的真实内存地址

**指定删除器**



注意事项：不要将一个原始指针构造成多个智能指针，因为这些智能指针会各自构造一个新的Ctlblock，而将一个智能指针赋值给另一个，二者的Ctlblock是相同的；不要在函数实参中创建智能指针，因为实参构造顺序不确定；避免循环引用；不要将unique和shared赋值混用

Weak\_ptr的lock()和expire()方法



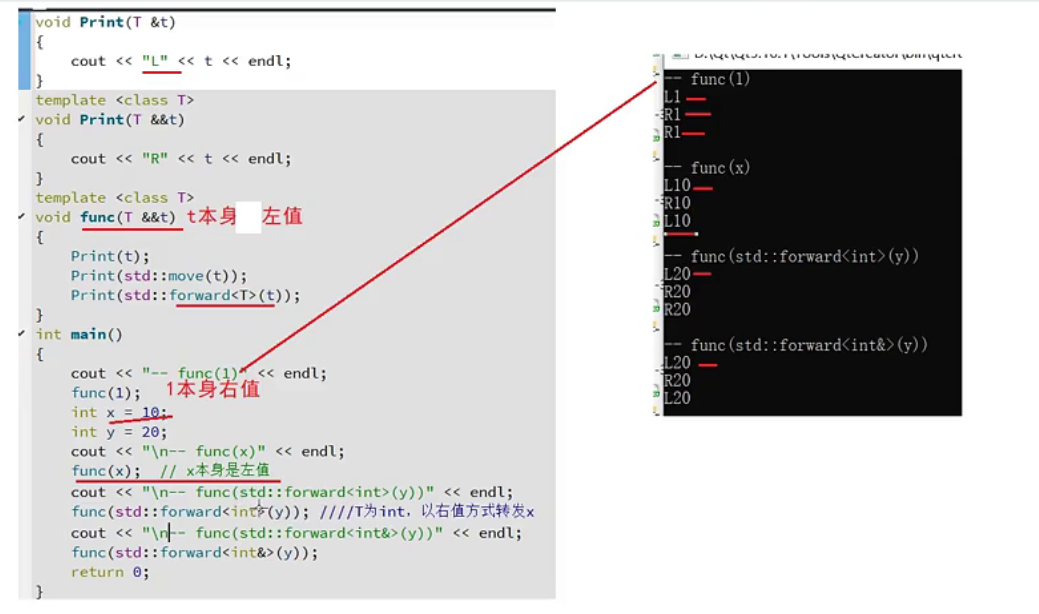
1. **右值引用与移动语义**



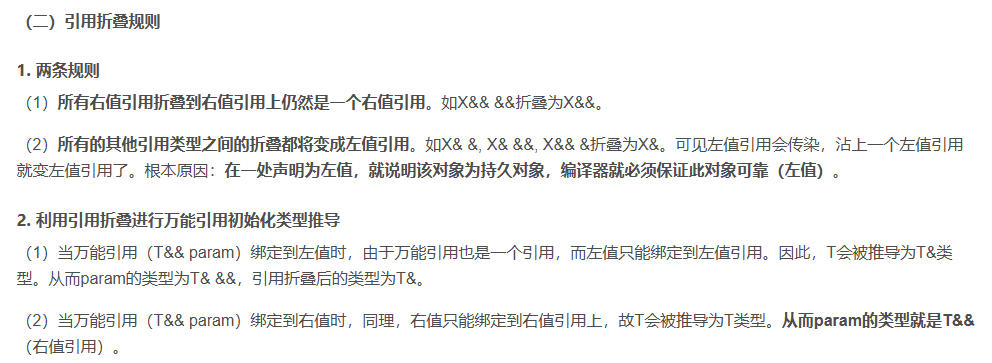
std::move会触发移动构造函数，默认情况下移动构造的优先级更高

移动构造、拷贝构造返回的是String，移动赋值、拷贝赋值返回的是String&

std::forward的作用是实现函数参数的完美转发，通俗的讲就是根据传入的参数，决定将参数以左值引用还是右值引用的方式进行转发。



引用折叠规则



forward即是为了使得参数保持其类型并进行转发





1. Lambda函数

捕获列表[]，在函数定义的时候即创建一份捕获参数的副本，如果是值传递则相当于创建一个新变量，如果是引用则创建一个新指针。

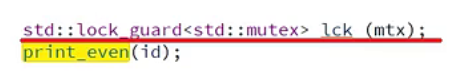
如果是空捕获列表，则匿名函数不能使用外层函数中生命的变量，此外还有[=]、[&]等简洁的值传递和引用传递。

1. std::thread

join()用于等待线程完成执行，确保线程按顺序执行；joinable()检查线程是否正在执行（是否可连接）;detach，使一个线程挂起（不可链接），成为守护进程。

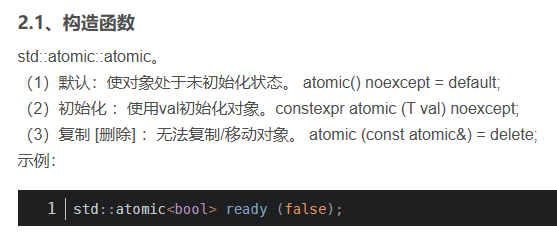
thread(func, 可变参数列表)，其中可变参数列表可以按顺序传入func需要的参数

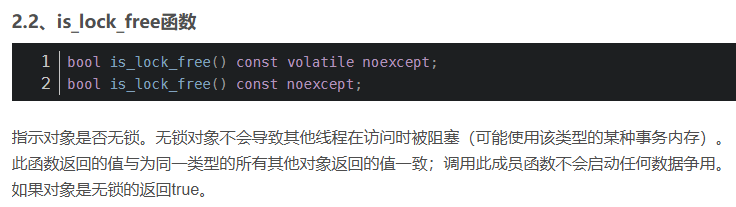
1. mutex



lck是栈上变量，退出作用域自动解锁，可以避免忘记解锁，此处lock\_guard可直接替换为unique\_lock

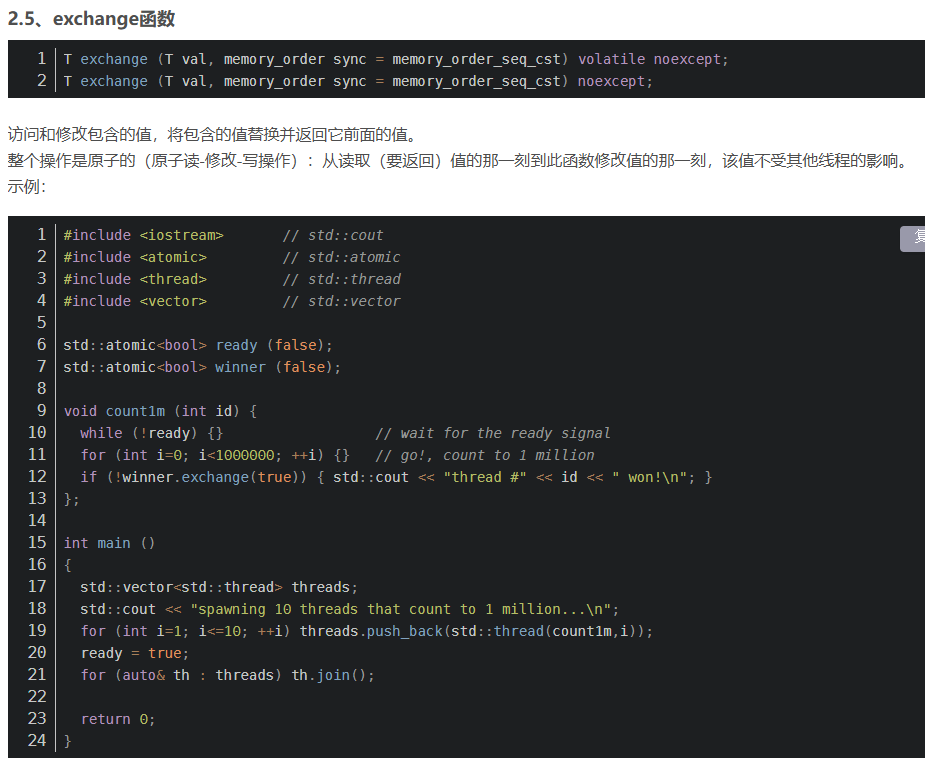
1. atomic



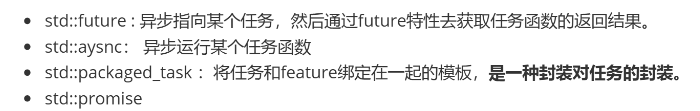




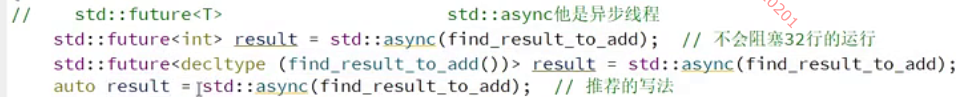




1. 异步操作

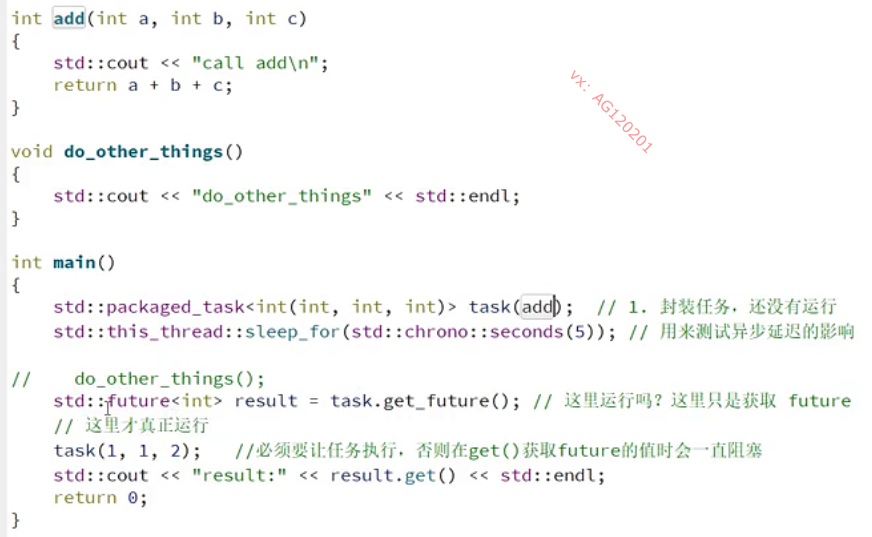


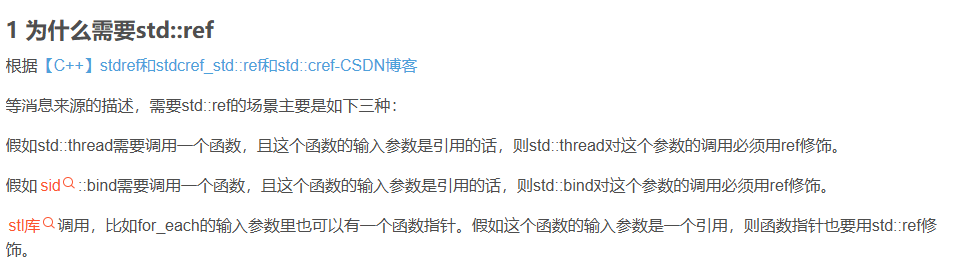
aysnc是异步线程，不会阻塞，只有调用get方法时才会阻塞，等待异步线程结束



以上是三种std::future的定义方法

Packaged\_task只是封装任务，当再次调用task时才开始运行





1. 函数回调



Bind两种使用方法：1.只绑定函数，手动传参；2.同时绑定参数和函数

参数列表占位符统一从下标1开始

需要重载函数时在函数前面使用函数指针修饰