<https://blog.csdn.net/sjc_0910/article/details/118861539?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522165553383616782391843236%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=165553383616782391843236&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-118861539-null-null.142%5Ev17%5Erank_v33,157%5Ev15%5Enew_3&utm_term=c%2B%2B%E5%A4%9A%E7%BA%BF%E7%A8%8B&spm=1018.2226.3001.4187>

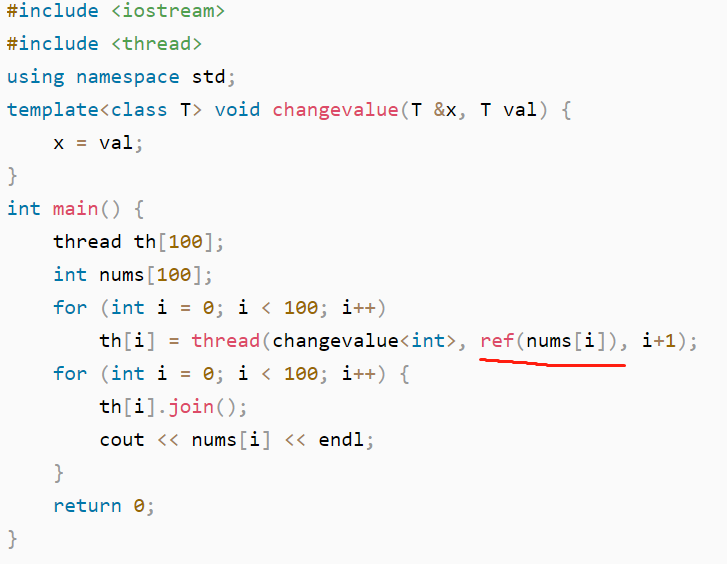
c++11 多线程

<https://blog.csdn.net/beidaol/article/details/89135277>

如何实现多线程

简单概念：





注意thread在传参时是以右值传递的，这里需要个左值转右值的操作

std::mutex 互斥锁

std::atomic 原子锁

C:\Users\Administrator\Documents\WeChat Files\siwangzhishen30\FileStorage\Temp\1656644954674.png

这就意味着即使是多线程，也要像同步进行一样**同步操作**atomic对象，从而省去了mutex上锁、解锁的时间消耗。

Std::async

Std::future



Ok，回到Games101作业7的多线程优化上来：

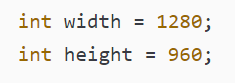
1. 明确多线程的划分

每个像素**发出主射线primary ray，并调用castRay()函数实现对像素着色**

对此进行划分

如我们逐行进行划分

已知scene大小即可算出每一线程包含的行数



int times = scene.height / thred ;

1. 创建执行函数（函数输入为每个线程最小行和最大行）
2. 解决多线程的资源争抢问题

上锁 mtx.lock()

4.分布进行调用以及join

