### [进程同步](http://onvip.com.cn/index.php/archives/3" \o "Permanent Link to 操作系统课程设计-银行家算法的实现)

**一、设计目的：**  
掌握进程同步。

**二、设计内容**

**实现读写锁。**

**三、开发环境**

+ Windows 10 64bits

+ Intel core i5-7300hq cpu@2.5GHz

+ gcc 7.1.0 x64bits-posix threads-seh (mingw7.1.0)

**四、分析设计**

如果一个任务有读写，且读写分离在多线程环境，那么就必须要用同步设施防止读写破坏文件流。最简单的设计是使用一个全局锁，每次读写前加锁，完成后解锁，这样虽然简单但是速度非常慢，不能享受多核处理器带来的性能提升，甚至较之同步版本还不如（因为加锁带来的penalty）。读写锁旨在解决该问题。它进一步细粒度锁，分为读锁，写锁。读锁运行多个读线程同时进行，写锁根据设计可以单个可以多个。

该实验实现了一个读者优先的读写锁，其中写线程同一时间只允许一个。

|  |
| --- |
|  |
| inline void wUnlock() { wmtx.unlock(); } |

inline void wLock() { wmtx.lock(); }

读锁逻辑是只要有一个读线程就加协锁

void RWLock::rLock() {

std::lock\_guard<std::mutex> guard(rmtx);

if(++rcount==1){

wmtx.lock();

}

}

读解锁逻辑是除非当前读线程为零否则不解锁

void RWLock::rUnlock() {

std::lock\_guard<std::mutex> guard(rmtx);

if(--rcount==0){

wmtx.unlock();

}

}

两个逻辑结合起来就是只要有>=1的读线程就锁上写互斥量，在这（多个读线程）期间无法执行写，直到没有读线程（readCount==0），此时解锁写互斥量，写线程可以竞争该锁。由于写锁总是要等到读线程完毕，这里的实现自然而然就是读优先了。

**五**. **运行示例及结果分析**

无

六、**心得与体会**

读写锁有很多种实现方法，可以使用，两个互斥量，也可以使用条件变量控制，这里使用了相对简单但是性能不友好的两互斥量。