线程的互斥和同步

临界资源概念：

不能同时访问的资源，比如写文件，只能由一个线程写，同时写会写乱。

比如外设打印机，打印的时候只能由一个程序使用。

外设基本上都是不能共享的资源。

生活中比如卫生间，同一时间只能由一个人使用。

必要性： 临界资源不可以共享

man手册找不到 pthread\_mutex\_xxxxxxx （提示No manual entry for pthread\_mutex\_xxx）的解决方法：

apt-get install manpages-posix-dev

互斥锁的创建和销毁

两种方法创建互斥锁，**静态方式**和**动态方式**

**动态方式：**

int pthread\_mutex\_init(pthread\_mutex\_t \*restrict mutex,const pthread\_mutexattr\_t \*restrict attr);

其中mutexattr用于指定互斥锁属性，如果为NULL则使用缺省属性。

**静态方式：**

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

锁的销毁：

int pthread\_mutex\_destroy(pthread\_mutex\_t \*mutex)

在Linux中，互斥锁并不占用任何资源，因此LinuxThreads中的 pthread\_mutex\_destroy()除了检查锁状态以外（锁定状态则返回EBUSY）没有其他动作。

互斥锁的使用：

int pthread\_mutex\_lock(pthread\_mutex\_t \*mutex)

int pthread\_mutex\_unlock(pthread\_mutex\_t \*mutex)

int pthread\_mutex\_trylock(pthread\_mutex\_t \*mutex)

vim 设置代码全文格式化：gg=G

读写锁

必要性：提高线程执行效率

特性：

写者：写者使用写锁，如果当前没有读者，也没有其他写者，写者立即获得写锁；否则写者将等待，直到没有读者和写者。

读者：读者使用读锁，如果当前没有写者，读者立即获得读锁；否则读者等待，直到没有写者。

注意：

同一时刻只有一个线程可以获得写锁，同一时刻可以有多个线程获得读锁。

读写锁出于写锁状态时，所有试图对读写锁加锁的线程，不管是读者试图加读锁，还是写者试图加写锁，都会被阻塞。

读写锁处于读锁状态时，有写者试图加写锁时，之后的其他线程的读锁请求会被阻塞，以避免写者长时间的不写锁

初始化一个读写锁 pthread\_rwlock\_init

读锁定读写锁 pthread\_rwlock\_rdlock

非阻塞读锁定　　 pthread\_rwlock\_tryrdlock

写锁定读写锁 pthread\_rwlock\_wrlock

非阻塞写锁定 pthread\_rwlock\_trywrlock

解锁读写锁 pthread\_rwlock\_unlock

释放读写锁 pthread\_rwlock\_destroy

死锁

概念：

避免方法：

1. 锁越少越好，最好使用一把锁
2. 调整好锁的顺序