1读写锁

读共享，写独占

2读写锁说明

读写锁与互斥量类似，不过读写锁的并行性更高。

读写锁可以有三种状态：

A 读模式加锁

B 写模式加锁

C 不加锁

3 在写加锁状态时，在解锁之前，所有试图对这个锁加锁的线程都会被阻塞。在读加锁状态时，所有试图以读模式对它进行加锁的线程都可以得到访问权限。但是如果线程希望以写模式加锁，它必须阻塞，直到所有的线程释放读锁。

4 读写锁很适合对于数据结构读的次数远大于写的情况。

5.相关函数

int pthread\_rwlock\_init(pthread\_rwlock\_t \*restrict rwlock, const pthread\_rwlockattr\_t \*restrict attr);

int pthread\_rwlock\_destroy(pthread\_rwlock\_t \*rwlock)  // 成功则返回0，失败则返回错误代码

int pthread\_rwlock\_rdlock(pthread\_rwlock\_t \*restrict rwlock) ;//读模式加锁

int pthread\_rwlock\_wrlock(pthread\_rwlock\_t \*restrict rwlock);//写模式加锁

int pthread\_rwlock\_unlock(pthread\_rwlock\_t \*restrick rwlock);

int pthread\_rwlock\_tryrdlock(pthread\_rwlock\_t \*restrict rwlock);

int pthread\_rwlock\_trywrlock(pthread\_rwlock\_t \*restrict rwlock);

int pthread\_rwlock\_trywrlock(pthread\_rwlock\_t \*restrict rwlock);

6 案例说明

|  |
| --- |
| #include<stdio.h>  #include<pthread.h>  #include<stdlib.h>  #include<unistd.h>  int counter;  //定义一把读写锁  pthread\_rwlock\_t rwlock;  //业务场景：3个线程不定时写同一全局资源，  //5个线程不定时读同一个全局资源  void \*th\_write(void \*arg) {  int t;  while(1) {  //上锁  pthread\_rwlock\_wrlock(&rwlock);  t = counter;  //休眠100微妙  usleep(100);  printf("write %x : counter = %d ++counter=%d\n",  (int)pthread\_self(),t,++counter);  //解锁  pthread\_rwlock\_unlock(&rwlock);  usleep(100);  }  }  void \*th\_read(void \*arg) {  while(1) {  pthread\_rwlock\_rdlock(&rwlock);  printf("read %x : %d\n",(int)pthread\_self(),counter);  pthread\_rwlock\_unlock(&rwlock);  usleep(100);  }  }  /\*特点是读时和写时都上锁\*/  int main(void) {  int i;  pthread\_t tid[8];  //初始锁  pthread\_rwlock\_init(&rwlock,NULL);  for(i = 0;i < 3;i++) {  pthread\_create(&tid[i],NULL,th\_write,NULL);  }  for(i = 0; i<5;i++) {  //从第四个开始  pthread\_create(&tid[i+3],NULL,th\_read,NULL);  }  pthread\_rwlock\_destroy(&rwlock);  for(i = 0;i< 8;i++) {  pthread\_join(tid[i],NULL);  }  return 0;  } |

通过给读写都加锁，让读写不会出现异常。

