


第8章 线程间的同步机制
——读写锁

主　讲：王小银

读写锁基本原理

- 读写锁把对共享资源的访问者分为读者和写者。
- 读写锁有三种状态：读模式下的加锁，写模式下的加锁，不加锁
- 读写锁操作主要分为两种情况：

（1）写者竞争到锁资源。

（2）读者竞争到锁资源。

初始化读写锁

函数名称	pthread_rwlock_init
函数功能	初始化读写锁
头文件	#include <pthread.h>
函数原型	int pthread_rwlock_init (pthread_rwlock_t *__restrict __rwlock, const pthread_rwlockattr_t *__restrict__attr);
参数	__restrict __rwlock：　指向要初始化的读写锁的指针。 __restrict__attr：　指向属性对象的指针，该属性对象定义要初始化的读写锁的特性，如果为NULL则使用默认属性。
返回值	0：　成功； 非0：　失败。

申请读锁——阻塞方式

函数名称	pthread_rwlock_rdlock
函数功能	以阻塞方式在读写锁上获取读锁（读锁定）。 如果没有写者持有该锁，并且没有写者阻塞在该锁上，则调用线程会获取读锁。如果调用线程未获取读锁，则该线程被阻塞直到它获取了该锁。
头文件	#include <pthread.h>
函数原型	int pthread_rwlock_rdlock (pthread_rwlock_t *__rwlock);
参数	__rwlock：　　读写锁指针。
返回值	0：　　成功； 非0：　　失败。

申请读锁——非阻塞方式

函数名称	pthread_rwlock_tryrdlock
函数功能	以非阻塞的方式来在读写锁上获取读锁。 如果有任何的读者持有该锁或有写者阻塞在该读写锁上，则立即失败返回。
头文件	#include <pthread.h>
函数原型	int pthread_rwlock_ tryrdlock (pthread_rwlock_t *__rwlock);
参数	__rwlock：　读写锁指针。
返回值	0：　成功； 非0：　失败。

申请写锁——阻塞方式

函数名称	pthread_rwlock_wrlock
函数功能	以阻塞方式在读写锁上获取写锁（写锁定）。 如果没有写者持有该锁，并且没有读者持有该锁，则调用线程会获取写锁。 如果调用线程未获取写锁，则该线程被阻塞直到它获取了写锁。
头文件	#include <pthread.h>
函数原型	int pthread_rwlock_wrlock (pthread_rwlock_t *__rwlock);
参数	__rwlock：　　读写锁指针。
返回值	0：　　成功； 非0：　　失败。

申请写锁——非阻塞方式

函数名称	pthread_rwlock_try wrlock
函数功能	以非阻塞的方式来在读写锁上获取写锁。 如果有任何的读者持有该锁或有写者阻塞在该读写锁上，则立即失败返回。
头文件	#include <pthread.h>
函数原型	int pthread_rwlock_try wrlock (pthread_rwlock_t *__rwlock);
参数	__rwlock：　　读写锁指针。
返回值	0：　　成功； 非0：　　失败。

释放读写锁

函数名称	pthread_rwlock_unlock
函数功能	释放读写锁
头文件	#include <pthread.h>
函数原型	int pthread_rwlock_unlock (pthread_rwlock_t *__rwlock);
参数	__rwlock：　　读写锁指针。
返回值	0：　　成功； 非0：　　失败。

中国大学MOOC

销毁读写锁

函数名称	pthread_rwlock_destroy
函数功能	用于销毁一个读写锁，并释放由 pthread_rwlock_init函数自动申请的相关资源
头文件	#include <pthread.h>
函数原型	int pthread_rwlock_destroy (pthread_rwlock_t *__rwlock);
参数	__rwlock：　　读写锁指针。
返回值	0：　　成功； 非0：　　失败。

示例程序

```
int main(void)
{   pthread_t ptd1, ptd2, ptd3, ptd4;
    pthread_rwlock_init(&rwlock, NULL);//初始化一个读写锁

    //创建线程
    pthread_create(&ptd1, NULL, fun1, NULL);
    pthread_create(&ptd2, NULL, fun2, NULL);
    pthread_create(&ptd3, NULL, fun3, NULL);
    pthread_create(&ptd4, NULL, fun4, NULL);

    //等待线程结束，回收其资源
    pthread_join(ptd1,NULL);
    pthread_join(ptd2,NULL);
    pthread_join(ptd3,NULL);
    pthread_join(ptd4,NULL);

    pthread_rwlock_destroy(&rwlock);//销毁读写锁
    return 0;
}
```

示例程序

```
#include<stdio.h>
#include<unistd.h>
#include<pthread.h>

pthread_rwlock_t rwlock; //读写锁
int num = 1;
void *fun1(void *arg)
{   while(1)
    {  pthread_rwlock_rdlock(&rwlock);
       printf("first read num:%d\n",num);
       pthread_rwlock_unlock(&rwlock);
       sleep(1);
    }
}
void *fun2(void *arg)
{   while(1)
    {  pthread_rwlock_rdlock(&rwlock);
       printf("second read num:%d\n",num);
       pthread_rwlock_unlock(&rwlock);
       sleep(2);
    }
}
```

```
void *fun3(void *arg)
{   while(1)
    {  pthread_rwlock_wrlock(&rwlock);
       num++;
       printf("write thread one\n");
       pthread_rwlock_unlock(&rwlock);
       sleep(1);
    }
}

void *fun4(void *arg)
{   while(1)
    {  pthread_rwlock_wrlock(&rwlock);
       num++;
       printf("write thread two\n");
       pthread_rwlock_unlock(&rwlock);
       sleep(2);
    }
}
```

谢谢大家！

中国大学MOOC