


第8章 线程间的同步机制

——互斥锁

互斥锁基本原理

- 互斥锁提供了对临界资源以互斥方式进行访问的同步机制，即以排他的方式防止临界资源被破坏。
- 互斥锁状态：“锁定”和“打开”两种
- 在使用临界资源时线程会去先申请互斥锁，如果此时互斥锁处于“打开”状态，则立刻占有该锁，将状态置为“锁定”。此时如果再有其他线程使用该临界资源时发现互斥锁处于“锁定”状态，则阻塞该线程，直到持有该互斥锁的线程释放该锁。

互斥锁的初始化

函数名称	pthread_mutex_init
函数功能	初始化互斥锁
头文件	#include <pthread.h>
函数原型	int pthread_mutex_init(pthread_mutex_t *restrict mutex, const pthread_mutexattr_t *restrict attr);
参数	restrict：需要被初始化的互斥锁指针； attr：指向描述互斥锁属性的指针。
返回值	0：成功； 非0：失败。

互斥锁的属性

属 性	含 义
PTHREAD_MUTEX_TIMED_NP	普通锁，当一个线程加锁后，其他线程无法获得锁，并形成等待队列
PTHREAD_MUTEX_RECURSIVE_NP	嵌套锁，允许一个线程对同一个锁多次加锁，并通过unlock解锁。如果是不同线程的请求，则在解锁时重新竞争
PTHREAD_MUTEX_ERRORCHECK_NP	检错锁，在同一线程请求同一锁时返回EDEADLK，否则执行的动作与类型PTHREAD_MUTEX_TIMED_NP相同
PTHREAD_MUTEX_ADAPTIVE_NP	适应锁，释放后重新竞争

申请互斥锁

函数名称	pthread_mutex_lock
函数功能	申请互斥锁
头文件	#include <pthread.h>
函数原型	int pthread_mutex_lock(pthread_mutex_t *mutex);
参数	mutex：　　指向互斥锁对象的指针。
返回值	0：　　成功； 非0：　　失败。

申请互斥锁

函数名称	pthread_mutex_ trylock
函数功能	申请互斥锁
头文件	#include <pthread.h>
函数原型	int pthread_mutex_ trylock (pthread_mutex_t *mutex);
参数	mutex：　　指向互斥锁对象的指针。
返回值	0：　　成功； 非0：　　失败。

互斥锁的销毁

函数名称	pthread_mutex_destroy
函数功能	销毁互斥锁
头文件	#include <pthread.h>
函数原型	int pthread_mutex_destroy(pthread_mutex_t *mutex);
参数	mutex：　　需要销毁的互斥锁对象的指针。
返回值	0：　　成功； 非0：　　失败。

释放互斥锁

函数名称	pthread_mutex_ unlock
函数功能	释放互斥锁
头文件	#include <pthread.h>
函数原型	int pthread_mutex_ unlock (pthread_mutex_t *mutex);
参数	mutex：　　指向互斥锁对象的指针。
返回值	0：　　成功； 非0：　　失败。

示例程序

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

pthread_mutex_t mutex;
int sum = 0;

typedef struct FuncArg
{
    int start;
    int end;
}FuncArg;
```

```
void *thread_handler(void *arg)
{
    int start = ((FuncArg *)arg)->start + 1;
    int end = ((FuncArg *)arg)->end;

    for (; start <= end; ++start)
    {
        pthread_mutex_lock(&mutex);     // 申请锁
        sum += start;                   // 临界区操作
        pthread_mutex_unlock(&mutex);  // 释放锁
    }

    return 0;
}
```

示例程序9.1


```
int main(int argc, char *argv[])
{
    pthread_t thids[4];
    FuncArg arg[4];

    pthread_mutex_init(&mutex, NULL);  // 初始化互斥锁

    int len = 10000 / 4;
     for (int i = 0; i < 4; ++i)
    {   arg[i].start = len * i;
        arg[i].end = len * (i + 1);
        pthread_create(&thids[i], NULL, thread_handler, &arg[i]);
    }
```

```
for (int i = 0; i < 4; ++i)
    {
        int *ret;
        pthread_join(thids[i], (void **)&ret);
    }

    pthread_mutex_destroy(&mutex);
    printf("sum = %d\n", sum);

    return EXIT_SUCCESS;
}
```

谢谢大家！