实验 1.3 自旋锁

```
#include <stdio.h>
#include <pthread.h>
// 定义自旋锁结构体
typedef struct {
 int flag;
} spinlock_t;
// 初始化自旋锁
void spinlock_init(spinlock_t *lock) {
 lock->flag = 0;
}
// 获取自旋锁
void spinlock_lock(spinlock_t *lock) {
 while (__sync_lock_test_and_set(&lock->flag, 1)) {
 // 自旋等待
 }
}
// 释放自旋锁
void spinlock_unlock(spinlock_t *lock) {
 __sync_lock_release(&lock->flag);
}
// 共享变量
int shared_value = 0;
// 线程函数
void *thread_function(void *arg) {
 spinlock_t *lock = (spinlock_t *)arg;

 for (int i = 0; i < 5000; ++i) {
 spinlock_lock(lock);
 shared_value++;
 spinlock_unlock(lock);
 }

 return NULL;
}
int main() {
 pthread_t thread1, thread2;
 pthread_attr_t attr;
pthread_attr_init(&attr);
 spinlock_t lock;
// 输出共享变量的值
printf("shared_value:%d\n",shared_value);
 // 初始化自旋锁
 spinlock_init(&lock);
 // 创建两个线程
 pthread_create(&thread1, &attr, thread_function, &lock);
 printf("thread1 create success!\n");
 pthread_create(&thread2, &attr, thread_function, &lock);
printf("thread2 create success!\n");
 // 等待线程结束
 pthread_join(thread1, NULL);
 pthread_join(thread2, NULL);
 // 输出共享变量的值
 printf("shared_value:%d\n",shared_value);
 return 0;
}
```