**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;

}