实验六 Linux线程的同步和互斥

**一、实验目的**

了解Linux下的线程和多线程编程，理解Linux下基于信号量的线程同步和互斥的实现，并实现生产者消费者同步问题。

**二、实验环境**

Ubuntu Linux操作系统

gcc编程环境

**三、实验要求**

跟着实验指导书步骤边做边思考，同时完成实验报告，将实验报告提交至学者网。

1. **实验内容与步骤：**

1、查阅Linux下线程、互斥锁、信号量有关函数和并理解参数含义

1）线程有关函数

#include <pthread.h>

创建线程函数：

int pthread\_create(

pthread\_t \*tid,

const pthread\_attr \*attr,

void\* (\*start\_routine)(void\*),

void \*arg);

线程主动退出函数：

void pthread\_exit(void \*retval\_ptr);

撤销线程函数：

int pthread\_cancel(pthread\_t id,

void \*retval\_ptr);

**2）互斥锁（互斥型信号量）有关函数**

初始化互斥锁（互斥型信号量）：

int pthread\_mutex\_init(

pthread\_mutex\_t \*mutex,

const pthread\_mutexattr\_t \*attr);

撤销互斥锁（互斥型信号量）：

int pthread\_mutex\_destroy(

pthread\_mutex\_t \*mutex);

申请互斥锁（互斥型信号量）：（加锁）

int pthread\_mutex\_lock(

pthread\_mutex\_t \*mutex);

释放互斥锁（互斥型信号量）：（解锁）：

int pthread\_mutex\_unlock(

pthread\_mutex\_t \*mutex);

3）信号量有关函数：

#include <semaphore.h>

信号量初始化：

int sem\_init(set\_t \*sem,int pshared,unsigned int value);

信号量的撤销：

int sem\_destroy(sem\_t \*sem)

请求（等待）信号量(P操作）：

int sem\_wait(sem\_t \*sem);

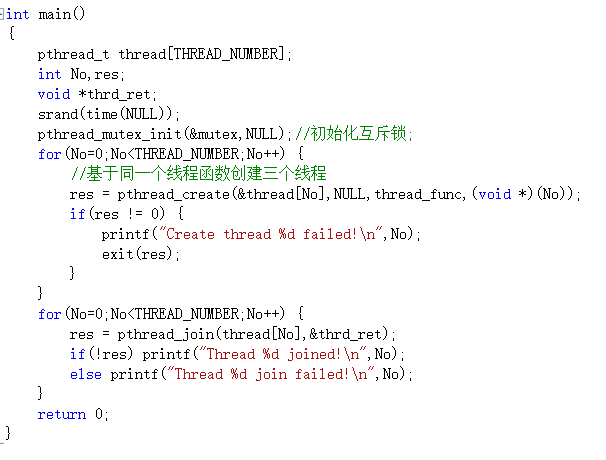
发送信号量（V操作）：

int sem\_post(sem\_t \*sem);

2、线程同步和互斥实验：

基于同一个线程函数创建3个线程，3个线程之间的并行执行。每个线程都有4次循环(可以看成4个小任务)，每次循环之间会随机等待1～10s的时间，意义在于模拟每个任务的到达时间是随机的，并没有任何特定的规律，如果没有采用互斥锁可以看到执行结果是无序的。要求在原有代码的基础上增加互斥锁功能，实现原本独立与无序的多个线程按顺序执行。





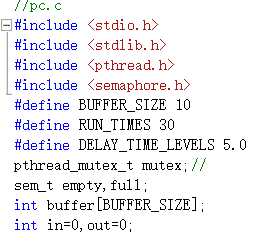
**编译：**gcc thread\_mutex.c -o thread\_mutex -lpthread

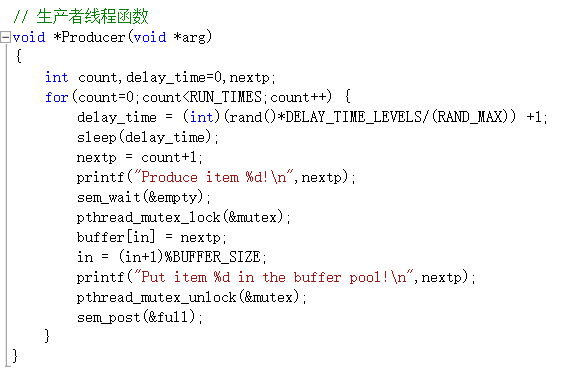
**执行：**./thread\_mutex

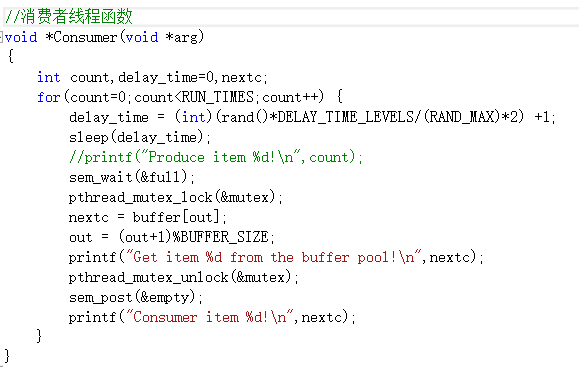
1）将实验结果写入实验报告；

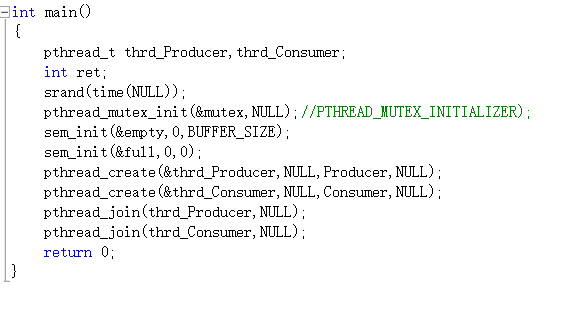
2）如果线程函数中去掉互斥锁加锁和解锁相关语句，执行结果如何？分析为什么？

3、经典生产者消费者问题（有多个缓冲区构成缓冲池，具体描述参考教科书）。









 编译：gcc pc.c -o pc -lpthread

 执行： ./pcr，注意这里要以root权限执行

1. 请将实验结果写入实验报告中。
2. 生产者线程的sem\_wait(&empty)和pthread\_mutex\_lock(&mutex)语句的次序能否调换？消费者线程sem\_wait(&full)和pthread\_mutex\_lock(&mutex)语句的次序能否调换？为什么？

3）如果希望有两个生产者和三个消费者，如何改写该同步演示程序？

提示：

创建线程函数：

int pthread\_create(

pthread\_t \*tid, // 线程标识符

const pthread\_attr \*attr, // 线程属性

void\* (\*start\_routine)(void\*), // 线程函数的起始地址

void \*arg); // 传递给start\_rountine的参数

以生产者为例，生产者函数中，void\* Producer(void\* arg)，可以通过arg的值（整型值）来区分不同的生产者。

arg 必须是void\*类型，所以需要在函数里面通过强制类型转换得到arg的值。在Producer函数开头添加以下语句：

int\* id = (int\*) arg;

将printf(“Produce item %d!\n, nextp); 改成

printf(“Producer %d produce item %d\n”, \*id, nextp); 来区分不同的生产者。

在主函数中，通过以下语句创建生产者线程、传递参数。

int producerID = 1;

pthread\_create(&thrd\_Producer, NULL, Producer, &producerID);