实验3 线程通信

**一、实验目的**

1. 掌握信号通信机制

2. 使用信号量机制实现经典线程同步的生产者和消费者问题

**二、实验工具与设备**

装有Linux系统的计算机。

**三、实验预备知识**

**1、线程预备知识  
pthreads定义了一套C程序语言类型、函数、与常量。以pthread.h和一个线程库实现。数据类型：**

**pthread\_t:**线程句柄

**pthread\_attr\_t:**线程属性

**线程操作函数：**

**pthread\_create():**创建一个线程

**pthread\_exit():**终止当前线程

**pthread\_cancel():**中断另外一个线程的运行

**pthread\_join():**阻塞当前的线程，直到另外一个线程运行结束

**pthread\_attr\_init():**初始化线程的属性

**pthread\_attr\_setdetachstate():**设置脱离状态的属性

**pthread\_attr\_getdetachstate():**获取脱离状态的属性

**pthread\_attr\_destroy():**删除线程的属性

**pthread\_kill():**向线程发送一个信号

**创建线程：**

**int pthread\_create(pthread\_t \*thread, const pthread\_attr\_t  \*attr, void \*(\*start\_rtn)  (void\*),void  \*arg);**

**\*thread:**线程id (指向线程标识符的指针)

**\*attr:**线程属性（通常为空）

**\*start\_rtn:**线程要执行的函数（线程运行函数的起始地址）

**\*arg:start\_rtn的参数**

**返回值为零才算成功，其它表示错误**

**编译时需要加上 -lpthread库**

**2、线程互斥锁预备知识**

linux下为了多线程同步，通常用到锁的概念。  
posix下抽象了一个锁类型的结构：ptread\_mutex\_t。通过对该结构的操作，来判断资源是否可以访问。顾名思义，加锁(lock)后，别人就无法打开，只有当锁没有关闭(unlock)的时候才能访问资源。

即对象互斥锁的概念，来保证共享数据操作的完整性。每个对象都对应于一个可称为" 互斥锁" 的标记，这个标记用来保证在任一时刻，只能有一个线程访问该对象。

使用互斥锁（互斥）可以使线程按顺序执行。通常，互斥锁通过确保一次只有一个线程执行代码的临界段来同步多个线程。互斥锁还可以保护单线程代码。

　　要更改缺省的互斥锁属性，可以对属性对象进行声明和初始化。通常，互斥锁属性会设置在应用程序开头的某个位置，以便可以快速查找和轻松修改。

 函数原型：

int pthread\_mutex\_init(pthread\_mutex\_t \*restrict mutex,  
          const pthread\_mutexattr\_t \*restrict attr);

该函数用于C函数的多线程编程中，互斥锁的初始化。

pthread\_mutex\_init()函数是以动态方式创建互斥锁的，参数attr指定了新建互斥锁的属性。如果参数attr为NULL，则使用默认的互斥锁属性，默认属性为快速互斥锁 。互斥锁的属性在创建锁的时候指定，在LinuxThreads实现中仅有一个锁类型属性，不同的锁类型在试图对一个已经被锁定的互斥锁加锁时表现不同。

pthread\_mutexattr\_init()函数成功完成之后会返回零，其他任何返回值都表示出现了错误。

函数成功执行后，互斥锁被初始化为锁住态。

**3、线程信号量预备知识**

 #include <semaphore.h>

int sem\_init(sem\_t \*sem, int pshared, unsigned int value);  
//初始化信号量  
       int sem\_wait(sem\_t \* sem);  
//等待信号,获取拥有权  
       int sem\_trywait(sem\_t \* sem);

       int sem\_post(sem\_t \* sem);  
//发出信号即释放拥有权  
       int sem\_getvalue(sem\_t \* sem, int \* sval);

       int sem\_destroy(sem\_t \* sem);  
//注销信号量,在linux中其本质是没有任何作用的，它不做任何事情。

**实验开始：**

**1.创建多个线程，模拟生产者和消费者问题的求解。**

**参考代码：**

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<pthread.h>

#include<semaphore.h>

#define PRODUCER\_NUM 5//生产者数目

#define CONSUMER\_NUM 5//消费者数目

#define POOL\_SIZE 11//缓冲池大小

int pool[POOL\_SIZE];//缓冲区

int head=0;//缓冲池读取指针

int rear=0;//缓冲池写入指针

sem\_t room\_sem;//同步信号信号量，表示缓冲区有可用间

sem\_t product\_sem;//同步信号量，表示缓冲区有可用品

pthread\_mutex\_t mutex;

void \*producer\_fun(void\*arg)

{

while(1)

{

sleep(rand()%10+1);

sem\_wait(&room\_sem);

pthread\_mutex\_lock(&mutex);

//生产者往缓冲池中写入数据

pool[rear]=1;

rear=(rear+1)%POOL\_SIZE;

printf("producer%d write topool\n",(int)arg);

printf("pool sizeis%d\n",(rear-head+POOL\_SIZE)%POOL\_SIZE);

pthread\_mutex\_unlock(&mutex);

sem\_post(&product\_sem);

}

}

void \*consumer\_fun(void\*arg)

{

while(1)

{

int data;

sleep(rand()%10+1);

sem\_wait(&product\_sem);

pthread\_mutex\_lock(&mutex);

//消费者从缓冲池读取数据

data=pool[head];

head=(head+1)%POOL\_SIZE;

printf("consumer%dreadf rom pool\n",(int)arg);

printf("poolsizeis%d\n",(rear-head+POOL\_SIZE)%POOL\_SIZE);

pthread\_mutex\_unlock(&mutex);

sem\_post(&room\_sem);

}

}

int main()

{

srand((int)time(0));

pthread\_t producer\_id[PRODUCER\_NUM];

pthread\_t consumer\_id[CONSUMER\_NUM];

pthread\_mutex\_init(&mutex,NULL);//初始化互斥量

int ret=sem\_init(&room\_sem,0,POOL\_SIZE-1);//初始化信号量room\_sem为缓冲池大小

if(ret!=0)

{

printf("sem\_initerror");

exit(0);

}

ret=sem\_init(&product\_sem,0,0);//初始化信号量product\_sem为0，开始时缓冲池中没有数据

if(ret!=0)

{

printf("sem\_initerror");

exit(0);

}

int i;

for(i=0;i<PRODUCER\_NUM;i++)

{

//创建生产者线程

ret=pthread\_create(&producer\_id[i],NULL,producer\_fun,(void\*)i);

if(ret!=0)

{

printf("producer\_iderror");

exit(0);

}

//创建消费者线程

ret=pthread\_create(&consumer\_id[i],NULL,consumer\_fun,(void\*)i);

if(ret!=0)

{

printf("consumer\_iderror");

exit(0);

}

}

for(i=0;i<PRODUCER\_NUM;i++)

{

pthread\_join(producer\_id[i],NULL);

pthread\_join(consumer\_id[i],NULL);

}

exit(0);

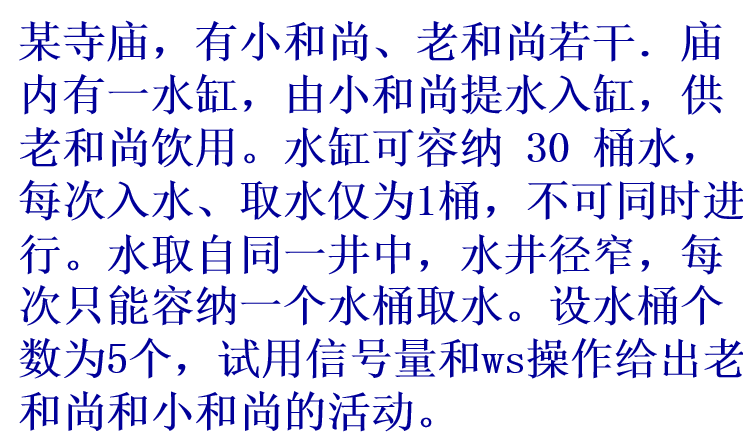
}

**编译注意事项**

**注意一定要在最后加入参数 –pthread**

**例如 gcc test.c –o test -pthread**

2．利用线程已经信号量，编写一个老和尚和小和尚抬水喝水的问题解决代码。



**五、实验总结**

1．写出实验报告。

2.总结解决生产者和消费者问题，关键的问题在哪儿，你是怎么解决的。

3、总结小和尚老和尚抬水喝水问题、