## 操作系统

## 第五次作业

**同步和互斥**

**6.4 Explain why spinlocks are not appropriate for single-processor systems yet are often used in multiprocessor systems.**

当一个进程处于自旋状态时，它将持续占用CPU资源并不断尝试获取锁。在单处理器系统中，由于自旋的进程对CPU的占用，拥有锁的进程可能因为无法得到CPU而等待，从而导致了两个进程的相互等待，造成系统的死锁。而在多处理器系统中，拥有锁的进程可以在其他未被占用的处理器上运行，不会造成系统死锁。所以自旋锁不适合单处理器系统，而经常被用于多处理器系统。

**6.9 Show that, if the wait() and signal() semaphore operations are not executed atomically, then mutual exclusion may be violated.**

在wait操作中，判断进程所需的资源是否被占用，若有被占用的资源则进程进入该资源的等待队列；在signal操作中，资源的等待队列中的第一个进程被唤醒。如果wait操作和signal操作不是原子地执行，则有可能某资源的队列中的进程被唤醒的同时，有其他进程也获取到资源的空闲状态而进入临界区，即资源有可能被分配多次。这违反了进程的互斥原则。

**6.11 The Sleeping-Barber Problem. A barbershop consists of a waiting room with n chairs and a barber room with one barber chair. If there are no customers to be served, the barber goes to sleep. If a customer enters the barbershop and all chairs are occupied, then the customer leaves the shop. If the barber is busy but chairs are available, the customer sits in one of the free chairs. If the barber is asleep, the customer wakes up the barber. Write a program to coordinate the barber and the customers.**

semaphore customer, barber, mutex; //信号量

int waiting, chairs; //等待顾客个数，椅子数

Barber:

while true {

P(customer);

P(mutex);

waiting = waiting – 1;

V(mutex);

//Cut hair

V(barber);

}

Customer:

P(mutex);

if (waiting < chairs){

waiting = waiting + 1;

V(customer);

V(mutex);

P(barber);

//Get hair cut

else{

V(mutex);

//Customer leave

}

1. **有一个系统，定义P、V操作如下：**

**P(s):**

**s:=s-1;**

**if s<0 then**

**将本进程插入相应队列末尾等待；**

**V(s) ：**

**s:=s+1;**

**if s<=0 then**

**从相应等待队列队尾唤醒一个进程，将其插入就绪队列；**

**问：**

**这样定义P、V操作是否有问题？**

**用这样的P、V操作实现N个进程竞争使用某一共享变量的互斥机制。**

有问题。因为这样定义的P、V操作是先进后出的，可能会出现进程无休止的等待，造成饥饿现象。

定义信号量数组 S[N-1] ，其中 S[i] = i ，对于每个进程执行操作：

void function(){

for(int i = n-1 ; i > 0 ; i--){

P(S[i]);

}

//Critical section

for(int i = 1 ; i < n ; i++){

V(S[i]);

}

//Things to do

}

**2. 第二类读者写者问题：写者优先**

**条件：**

**多个读者可以同时进行读**

**写者必须互斥（只允许一个写者写，也不能读者写者同时进行）**

**写者优先于读者（一旦有写者，则后续读者必须等待，唤醒时优先考虑写者）**

semaphore write, read, wcountMutex, rcountMutex; //信号量

int wcount, rcount; //写进程与读进程计数

Write:

P(wcountMutex);

if (wcount==0) P(read);

wcount++;

V(wcountMutex);

P(write);

//write

V(write);

P(wcountMutex);

wcount--;

if(wcount==0)

V(read);

V(wcountMutex);

Read:

P(read);

V(read);

P(rcountMutex);

if(rcount==0) P(write);

rcount++

V(rcountMutex);

//read

P(rcountMutex);

rcount--

if(rcount==0) V(write);

V(rcountMutex);

**3. 把学生和监考老师都看作进程，学生有 N 人，教师 1 人。考场门口每次只能进出一个人，进考场原则是先来先进。当 N 个学生都进入考场后，教师才能发卷子。学生交卷后可以离开考场。教师要等收上来全部卷子并封装卷子后才能离开考场。**

**问：**

**共需设置几个进程？**

**试用P、V操作解决上述问题中的同步和互斥关系。**

N+1个进程，分别代表N个学生和1个老师。

semaphore door, s1, s2, s3; //信号量

int student; //不在场的学生数量

Student:

P(door);

//Go into room

V(door);

P(s1);

student = student – 1;

if (student == 0) V(s2);

V(s1);

//Exam

P(s1);

student = student + 1;

//Hand in paper

if (student == N) V(s3);

V(s1);

P(door);

//Go out of room

V(door)

Teacher:

P(door);

//Go into room

V(door);

P(s2);

//Give away papers

//Exam

//Collect papers

P(s3);

P(door);

//Go out of room

V(door);