第四次作业

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，进程检查所需的资源是否已被占用，若资源已被占用，进程会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会被唤醒并分配资源。如果 wait 和 signal 操作不是原子地执行，就可能出现这样一种情况：当某个资源的等待队列中的进程被唤醒的同时，其他进程也能检测到资源处于空闲状态并进入临界区。这样，资源可能会被多个进程同时获取，违反了互斥原则。

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.

Barber：

```
while (true) {
    P(customers);
    P(mutex);
    waiting = waiting - 1;
    V(barber);
    V(mutex);
    cut_hair();
}
```

Customer：

```
P(mutex);
if (waiting < n) {
```

```
        waiting = waiting + 1;
        V(customers);
        V(mutex);
        P(barber);
        get_haircut();
} else {
        V(mutex);
}
```

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。这样的 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。

```
// 信号量数组 S[N-1] 初始时 S[i] = 1;
void function() {
    // 进入临界区前，按顺序执行 P 操作
    for (int i = n-1; i >= 0; i--) {
        P(S[i]);
    }
    // 临界区
    // Critical section
    // 离开临界区后，按顺序执行 V 操作
    for (int i = 0; i < n; i++) {
        V(S[i]);
    }
```

```
        // 临界区外的其他操作
        // 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）

```
int wcount = 0, rcount = 0;
semaphore write = 1;
semaphore read = 1;
semaphore wcountMutex = 1;
semaphore rcountMutex = 1;
// 写者
void Write() {
    P(wcountMutex);
    if (wcount == 0) {
        P(read);
    }
    wcount++;
    V(wcountMutex);
    P(write);
    // 执行写操作
    write_data();
    V(write);
    P(wcountMutex);
    wcount--;
    if (wcount == 0) {
        V(read);
    }
    V(wcountMutex);
}
// 读者
void Read() {
    P(read);
    V(read);
    P(rcountMutex);
    if (rcount == 0) {
        P(write);
    }
```

```
        rcount++;
        V(rcountMutex);
        // 执行读操作
        read_data();
        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。

共需设置 N+1 个进程，包含 N 个学生进程和 1 个老师进程。

```
void student(int id) {
        P(mutex);
        // 学生进入考场
        studentCount++;
        if (studentCount == N) {
            // 教师发卷
            V(allIn);
        }
        V(mutex);
        // 学生交卷
        P(mutex);
        doneCount++;
        if (doneCount == N) {
            V(allDone);
        }
        V(mutex);
        // 学生离开考场
}
void teacher() {
        P(allIn);
```

```
// 教师发卷
P(allDone);
// 教师收卷并封装
}
```