

第六章 进程管理

例题

例 6.1 某车站售票厅，任何时刻最多可容纳 200 名购票者进入，当售票厅中少于 200 名购票者时，则厅外的购票者可立即进入，否则需在外面等待。若把一个购票者看作一个进程，请回答下列问题：

(1) 试问：应编制几个程序和设置几个进程？程序和进程的对应关系如何？

(2) 试用 P、V 操作编写购票者进程的同步算法。

解：(1) 应编制 1 个程序，设置 n 个进程，其中每名购票者就是一个进程。多个进程共享同一程序，即程序：进程=1： n 。

(2)

```
semaphore s;
```

```
s.value=200;
```

```
cobegin
```

```
    repeat  $B_i$ ;
```

```
coend
```

```
Process  $B_i$ 
```

```
{    .....;
```

```
    P(&s);
```

```
    进入大厅购票;
```

```
    购完票离开大厅;
```

```
    V(&s);
```

```
}
```

例 6.2，打瞌睡的理发师问题

1. 问题描述

理发店有一名理发师，一把理发椅，还有 N 把供等候理发的顾客坐的普通椅子。如果没有顾客到来，理发师就坐在理发椅上打瞌睡。当顾客到来时，就唤醒理发师。如果顾客到来时理发师正在理发，顾客就坐下来等待。如果 N 把椅子都坐满了，顾客就离开该理发店去别处理发，如图 6-1 所示。要求为理发师和顾客各编写一段程序，描述他们的行为，并用

信号量保证上述过程的实现。

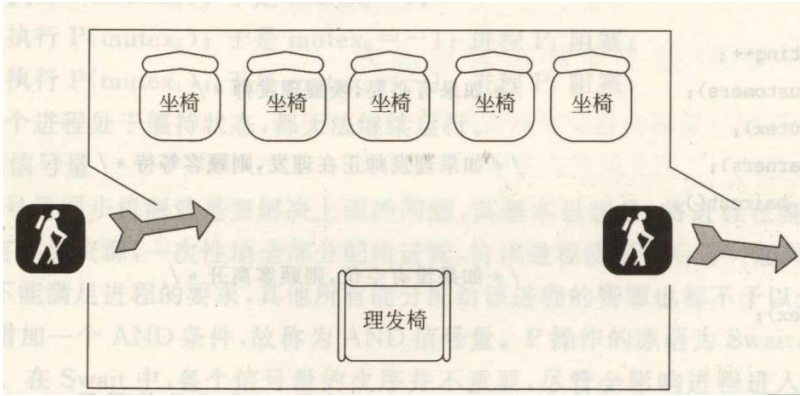


图 6-1 打瞌睡的理发师问题

2. 用信号量解决打瞌睡的理发师问题

为理发师和顾客分别写一段程序，并创建进程。理发师开始工作时，先看看店里有无顾客，如果没有，则在理发椅上打瞌睡；如果有顾客，则为等待时间最长的顾客理发，且等待人数减 1。顾客来到店里，先看看有无空位，如果没有空位，就不等了，离开理发店；如有空位，则等待，等待人数加 1；如果理发师在打瞌睡，则将其唤醒。

为了解决上述问题，设一个计数变量 `waiting`，表示等候理发的顾客人数，初值为 0；设三个信号量：`customers` 用来记录等候理发的顾客数(不包括正在理发的顾客)；`barbers` 用来记录正在等候顾客的理发师数(其值为 0 或 1)；`mutex` 用于互斥。程序描述如下。

```
#define CHAIRS 5      /*为等候的顾客准备的坐椅数*/

semaphore customers=0;

semaphore barbers=0;

semaphore mutex=1;

int waiting;

void barber()        /*理发师进程*/
{
    while (true)
    {
        P(customers);    /*如果没有顾客，理发师就打瞌睡*/
        P(mutex);        /*互斥进入临界区*/
        Waiting--;
        V(barbers);       /*理发师准备理发了*/
    }
}
```

```

        V(mutex);

        cut-hair();      /*理发*/
    }
}

void customer()      /*顾客进程*/
{
    P(mutex);

    if (waiting<CHAIRS)    /*如果有空位，则顾客等待*/
    {
        waiting++;

        V(customers);      /*如果有必要，唤醒理发师*/

        V(mutex);

        P(barners);      /*如果理发师正在理发，则顾客等待*/

        get-haircut();

    }

    else    /*如果没有空位，则顾客离开*/

        V(mutex);

}

```

当有一个顾客来到理发店时，执行 `customer()` 过程，首先获取信号量 `mutex` 进入临界区，如果不久另一个顾客到来，新到顾客只能等到释放 `mutex` 后才能进入。

进入临界区的顾客随后查看是否有椅子可坐，若没有，则释放 `mutex` 并离开；若有椅子可坐，则对计数变量加 1，之后执行 `V(customers)` 操作唤醒理发师。当顾客释放 `mutex` 后，理发师获得 `mutex`，他进行一些准备后开始理发。理发完毕，顾客退出 `customer()` 过程，离开理发店。

3. 思考问题

- (1) 为什么理发师进程中使用循环语句，而顾客进程却没有？
- (2) 程序中 `waiting` 的计数作用能否用信号量 `customers` 代替？

例 6.3 设系统中有三类资源 `R1`、`R2`、`R3` 和 `R4`，又设系统中有 5 个进程 `P1`，`P2`，`P3`，`P4` 和

P5。在 T0 时刻系统状态如下：

资源 进程	最大需求量				已分配资源量				剩余资源量			
	R1	R2	R3	R4	R1	R2	R3	R4	R1	R2	R3	R4
P1	8	6	4	1	1	2	1	1	2	1	1	3
P2	4	3	3	1	3	1	1	1				
P3	10	1	3	2	4	1	3	2				
P4	3	3	3	1	3	2	2	0				
P5	5	4	6	3	1	1	3	1				

(1) 系统是否处于安全状态？如果是，给出安全序列。如果不是，说明理由。

(2) 如果进程 P5 申请资源为(1, 1, 1, 2)，能否实施分配？为什么？

解：安全的。

先计算各个进程还需要的资源：

P1: (7 4 3 0), P2:(1 2 2 0), P3: (6 0 0 0), P4:(0 1 1 1), P5: (4 3 3 2);

系统现在可用资源为(2 1 1 3)，此时可满足 P4 资源的最大需求，将资源分配给 P4，P4 执行完并归还它所占的全部资源后系统可用资源为(6 2 4 5)，此时可满足 P2 或 P3 对资源的最大需要，先分配给 P2，待 P2 执行完毕后再分配给 P3，待 P3 执行完毕系统收回所占的全部资源，此时系统有可用资源为(10 5 6 7)，此时可满足 P1 或 P5 对资源的最大需要。因此系统是安全的，安全执行序列有 P4 P2P3 P1P5 或 P4 P3P2 P5P1 等

(2) 在 T0 时刻，若进程 P5 申请资源为(1, 1, 1, 2)，虽然系统现有可用资源(2 1 1 3)满足申请需要，但不满足 P5 的最大资源需求(4 3 3 2)，按银行家算法只有在满足最大资源需求时才给予分配，因此不分配；如果分配，那么分配后系统剩余资源为(1 0 0 1)满足不了任何一个进程对资源的最大需要，因此不能分配。

习题：

1. 什么叫并发进程的执行产生与时间有关的错误？这种错误表现在哪些方面？试举例说明之。
2. 什么叫临界区？对临界区的管理应符合哪些原则？
3. 在信号量 s 上作 P、V 操作时，s 的值会发生变化，当 s 的值大于 0，s 的值等于 0，s 的值小于 0 时，其物理意义各是什么？
4. 设有 N 个进程，共享一个资源 R，但每个时刻只允许一个进程使用 R。算法如下：设置一个整型数组 flag[N]，其每个元素对应表示一个进程对 R 的使用状态，若为 0 表

示该进程不在使用 R，为 1 表示该进程要求或正在使用 R，所有元素的初值均为 0。

```
process Pi
{
    ...
    flag[i] = 1;
    for (j=0; j<i; j++)
        do while (flag[j]);
    for (j=i+1; j<N; j++)
        do while (flag[j]);
    use resource R ;
    flag[i] = 0;
    ...
}
```

试问该算法能否实现上述功能？为什么？若不能请用 P、V 操作改写上述算法。

5. 有三个进程 R, M, P, R 负责从输入设备读入信息并传送给 M, M 将信息加工并传送给 P, P 将打印输出，写出下列条件下的并发进程程序描述。

(1) 一个缓冲区，其容量为 K；

(2) 两个缓冲区，每个缓冲区容量均为 K。

6. 假定一个阅览室最多可以容纳 100 人阅读,读者进入和离开阅览室时,都必须在阅览室门口的一个登记表上注册或注销。假定每次只允许一个人注册或注销,设阅览室内有 100 个座位。

(1) 试问：应编制几个程序和设置几个进程？程序和进程的对应关系如何？

(2) 试用 P、V 操作编写读者进程的同步算法。

7. 系统有输入机和打印机各一台，有两进程都要使用它们，采用 P、V 操作实现请求使用和归还释放后，还会产生死锁吗？若不会，说明理由；若会，你认为应怎样来防止死锁。

8. 系统中有 5 个资源被 4 个进程所共享，如果每个进程最多需要 2 个这种资源，试问系统是否会产生死锁？

9. 若系统有同类资源 m 个，被 n 个进程共享，问：当 $m > n$ 和 $m \leq n$ 时，每个进程最多可以请求多少个这类资源，使系统一定不会发生死锁？

10. 设系统有某类资源共 12 个，用银行家算法判断下列每个状态是否安全。如果是安全的，说明所有进程是如何能运行完毕的。如果是不安全的，说明为什么可能产生死锁。

状态 A

进程	占有资源数	最大需求
进程 1	2	6
进程 2	4	7
进程 3	5	6
进程 4	0	2

状态 B

进程	占有资源数	最大需求
进程 1	4	8
进程 2	2	6
进程 3	5	7