南京大学
NANJING UNIVERSITY

计算机操作系统

6 并发程序设计 – 6.3 PV操作

6.3.1 PV操作与进程互斥

理解信号量的概念

掌握记录型信号量

掌握PV操作原语

应用PV操作解决进程互斥

问题的提出

- TS或swap指令管理临界区，采用忙式轮询，效率低
- 关开中断管理临界区，不便交给用户程序使用

- 参考：操作系统访问硬件资源时采用“请求-等待-中断恢复”方式

信号量的构思

- 一种可动态定义的软件资源：信号量
  - 核心数据结构：等待进程队列
  - 信号量声明：资源报到，建立队列
  - 申请资源的原语：若申请不得，调用进程入队等待
  - 归还资源的原语：若队列中有等待进程，需释放
  - 信号量撤销：资源注销，撤销队列

记录型信号量的定义

- 记录型信号量：一种带数值的软资源

```
typedef struct semaphore {
        int value;              // 信号量值
        struct pcb *list; // 信号量等待进程队列指针
}
```

- 每个信号量建立一个等待进程队列
- 每个信号量相关一个整数值
  - 正值表示资源可复用次数
  - 0值表示无资源且无进程等待
  - 负值表示等待队列中进程个数

P操作原语与V操作原语

```
procedure P(semaphore:s) {
   s = s – 1;              //信号量减去1
   if (s < 0) W(s);    //若信号量小于0，则调用进程
                            被置成等待信号量s的状态
}
procedure V(semaphore:s) {
   s := s + 1;             //信号量加1
   if (s <= 0) R(s);  //若信号量小于等于0，则释放
                            一个等待信号量s的进程
}
```

PV操作解决进程互斥问题框架

```
semaphore s;
s = 1;
cobegin
   process Pi {
            ……
            P(s);
            临界区;
            V(s);
            ……
   }
coend;
```

PV操作解决机票问题

```
Int A[m];
Semaphore s;
s = 1;
cobegin
process Pi {
        int Xi;
        Li:按旅客定票要求找到A[j];
        P(s);
        Xi = A[j];
        If (Xi>=1) { Xi=Xi-1; A[j]=Xi; V(s); 输出一张票;
        } else { V(s); 输出票已售完;}
        goto Li;
}
coend
```

只有相同航班的票数才是相关的临界资源，所以用一个信号量处理全部机票会影响进程并发度

P操作与V操作在执行路径上一一匹配

PV操作解决机票问题（改进）

```
Int A[m];
Semaphore s[m];
For (int j=0;j<m;i++) s[j] = 1;
cobegin
process Pi {
        int Xi;
        L1:按旅客定票要求找到A[j];
        P(s[j]);
        Xi = A[j];
        If (Xi>=1) { Xi=Xi-1; A[j]=Xi; V(s[j]); 输出一张票;
        } else { V(s[j]); 输出票已售完; }
        goto L1;
}
coend;
```