信号量机制讨论

● 邓 莉

在操作系统中，进程管理是其重要内容之一。实施进程管理面临的主要问题是如何实现并发进程的同步。常用的同步机制有：信号量机制和管程机制。在这里，我们仅讨论信号量机制。

一、信号量的概念及其分类

信号灯是铁路交通管理中的一种常用设备，交通管理人员利用信号灯的状态（颜色）实现交通管理。操作系统中使用的信号量正是从交通管制中引用过来的一个术语。

信号量是由荷兰的计算机科学家 Dijkstra 于 1965 年提出的最早的同步方法。所谓信号量是一个仅能由同步原语对其进行操作的整型变量。Dijkstra 将这两个同步原语命名为“P 操作”和“V 操作”（P、V 来源于荷兰文的“发信号”和“等待”二词的第一个字母）。

信号量按其用途可分为：

(1)互斥信号量：对应着某一临界资源，其初值均为 1。

(2)同步信号量：对应着某一类共享资源，其初值为该共享资源类最初可用资源数目。

二、信号量的操作

P、V 操作是对信号量进行的原语操作（不可中断）。定义如下：

P(S)：信号量 S 减去 1，若结果小于 0，则调用 P(S)的进程被置成等待信号量 S 的状态。

V(S)：S 加 1，若结果不大于 0，则释放一个等待信号量 S 的进程。

程序描述如下：

```
procedure P(Var S:semaphore);
begin
    S:=S-1;
    if S<0 then W(S)
end;{p}
procedure V(Var S:semaphore);
begin
    S:=S+1;
    if S≤0 then R(S)
end;{V}
```

其中，W(S)，R(S)均为系统提供的过程。W(S)的功能是把调用过程 W 的进程置成等待信号量 S 的状态，并将进程入相应的等待队列；R(S)的功能是释放一个等待信号量 S 的进程。

S>0 时，其值表示某类资源的可用资源数，每执行一次 P 操作意味着请求分配一个单位的该类资源给执行 P 操作的进程，从而 S:=S-1。

S≤0 时，表明已经没有此类资源可分配，因此请求资源的进程将被阻塞在相应的信号量 S 的等待队列中。此时，S 的绝对值等于在该信号量上等待的进程数。执行一次 V 操作意味着进程释放出一个单位的该类可用资源，故 S:=S+1。S≤0 也表明信号量等待

队列中有因请求资源而被封锁的进程。因此，当有进程归还该类资源时，应唤醒等待队列中的一个进程，即改进程的阻塞状态为就绪状态，并插入就绪队列中。

三、信号量机制的应用

信号量机制是一种良好的同步工具。一般同步问题可以分成两类：一类是保证共享缓冲区（或共享数据）的合作进程的同步；另一类是保证一组合作进程按逻辑需要所确定的执行顺序。

1. 对于第一种情形，需设置的信号量数目由与同步现象相关的共享资源类型数目而定。其初值为各类资源最初可利用数目。程序描述中，在共享资源前，加入对应信号量的P操作；归还资源后，加入对应信号量的V操作。

一个典型的例子是生产者和消费者问题，即生产者生产物品并存入共享的缓冲器，供消费者取走使用。消费者从缓冲区内取走物品去消费。两种操作若不协调，会导致两种不正确的现象。一种是消费者从一个空缓冲器内取物品，另一种是生产者把物品存入已满的缓冲器。

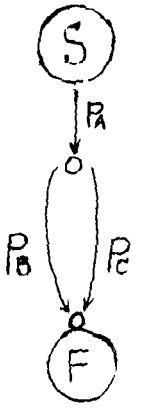
分析：生产者和消费者的同步源于对缓冲器、物品的正确操作，因此可分别设定 S_1、S_2 对应缓冲器和物品。初始时，生产者还未开始生产，物品数为0，缓冲区为空即可利用状态，于是，令 $S_1:=1, S_2:=0$。

程序描述如下：

```
B:integer;
S1,S2:semaphore;
S1:=1;        S2=0;
cobegin
    procedure producer;
    begin
        produce a product;
        P(S1);
        B:=product;
        V(S2);
    end;
    procedure consumer;
    begin
        P(S2);
        take a product;
        V(S1);
        consume
    end;
coend;
```

2. 对于第二种情形，常常是一对执行时序关系对应一个信号量，且初值为0。程序描述时，时序关系中先执行的进程尾部加入对应信号量的P操作，而后执行的进程首部加入V操作。

例　P_A, P_B, P_C 为一组合作进程，其流程图如下图所示，试用信号量实现这三个进程的同步。



分析　右图说明 P_A 执行结束后 P_B, P_C 才可开始执行，所以存在两对时序关系 $P_A \rightarrow P_B$，$P_A \rightarrow P_C$。

设两个同步信号量 S_b, S_c 分别表示进程 P_B 和 P_C 能否开始执行，初值均为0。

程序描述如下：

```
        ⋮
Sb:=0;
Sc:=0;
COBEGIN
    PA:BEGIN
        ⋮
        V(Sb);
        V(Sc)
    END;
    PB:BEGIN
        P(Sb);
        ⋮
```

END;

P_A:BEGIN

P(S_c);

⋮

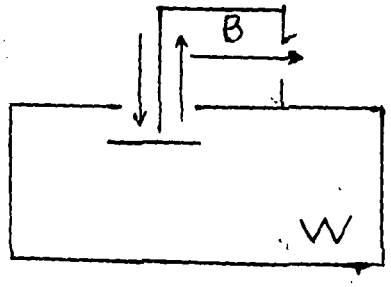
END;

COEND;

实际上，两种同步现象是统一的，对于时序关系的同步，我们可以认为它是由于一对时序关系中先执行进程结果数据作为后执行进程的源数据而造成的。那么，这部分共享数据就成为两个并发进程同步的根源。初始时，进程都未执行，这一部分数据不存在。因此，对应于该数据的信号量的初值为 0。

由此，对同步实例的分析，可按以下步骤进行：先识别互斥，同步现象；然后根据具体情况设定信号量，并赋初值；最后在各并发进程中适当位置添加对信号量的 P、V 操作。

例　一个理发店，由一间等候室 W 和一间工作室 B 组成。顾客可以从外面大街上进入 W，等候理发。两个房间与入口是并排的，且共享一扇日本式可滑动的推拉门（门总是挡住一个入口）。顾客在工作室内理完发，可



由 B 的旁门出去。（如图中箭头方向）。W 中有 N 把椅子，顾客必须坐着等候。理发师可由门上小窗查看，W 中无人就睡觉。否则开门，并叫一位顾客入内理发。顾客每进入一位，都拉铃通知理发师。试问：若把顾客和理发师都视为进程，请用 P、V 操作写出进程的同步算法。

分析：该实例中互斥资源只有一个——门，可设定一个互斥信号变量 mutex，其初值为 1。

同步现象有两个：一个是 W 中有空椅子时，顾客方可进入 W，二是 W 中有人等候时，理发师才工作。因此，分别设信号量 Sn，S，其初值分别为 n，0。

进程描述如下：

mutex:=1;

S_n:=n;　　S:=0;

顾客进程：

↓

P(Sn);

P(mutex);

推门进入 W;

V(mutex);

↓

V(S);

坐下等候;

↓

被理发师叫入 B

V(Sn)

↓

退出

理发师进程

↓

P(S);

↓

P(mutex);

开门叫进一位顾客

V(mutex);

↓

理发

（作者单位：湖北大学数学与计算机学院计算机系）