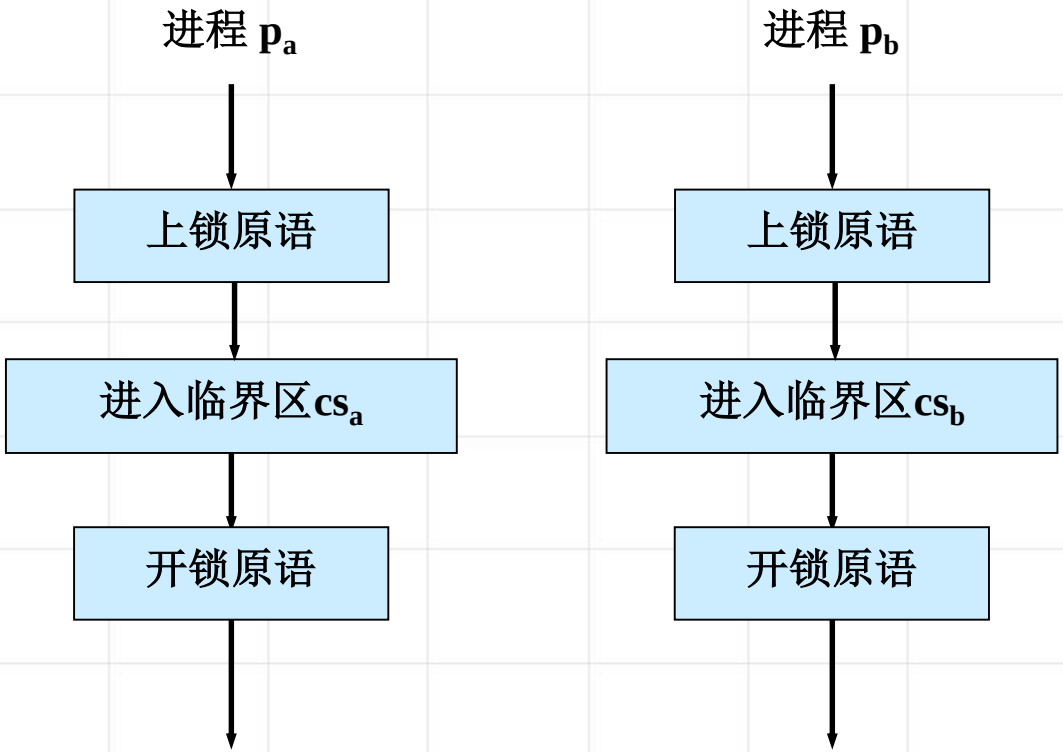
第4章
进程及进程管理

进程互斥与同步的实现

1. 用上锁原语和开锁原语实现进程互斥

(1) 框图描述



两个进程利用上锁、开锁原语实现互斥

(2) 程序描述

程序 task1

```
main( )
{
   int w=1;        /* 互斥锁 */
   cobegin
     pa( );
     pb( );
   coend
}
```

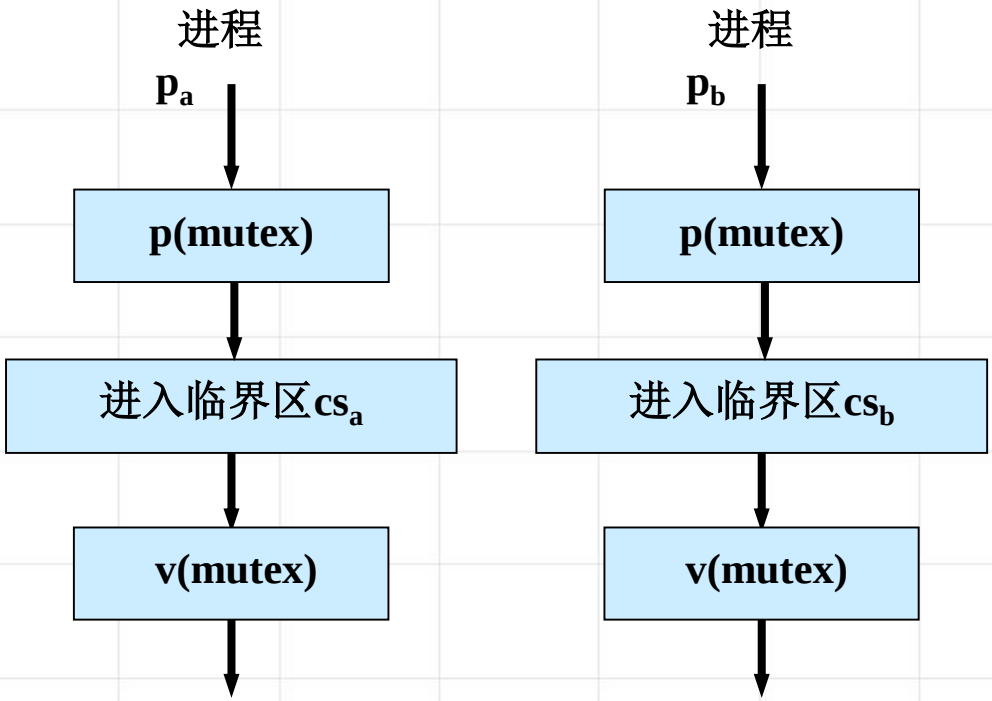
```
pa( )
{
  M
   lock(w);
     csa ;
   unlock(w);
    M
}
```

```
pb( )
{
  M
   lock(w);
     csb ;
   unlock(w);
    M
}
```

2. 用信号灯的P、V操作实现互斥

(1) 框图描述

设：mutex为互斥信号灯，初值为1。



两个进程利用信号灯的P、V操作实现互斥

(2) 程序描述

程序 task2

```
main( )
{
   int mutex=1;        /* 互斥信号灯 */
   cobegin
      pa( );
      pb( );
   coend
}
pa( )                          pb( )
{                              {
        M                              M
    p(mutex);                      p(mutex);
      csa ;                          csb ;
    v(mutex);                      v(mutex);
        M                              M
}                              }
```

(3) 信号灯可能的取值

两个并发进程，互斥信号灯的值仅取1、0和－1三个值。

mutex=1

表示没有进程进入临界区；

mutex=0

表示有一个进程进入临界区；

mutex=－1

表示一个进程进入临界区，另一个进程等待进入。

(4) 例

x代表某航班机座号，p_a和p_b两个售票进程，售票工作是对变量x加1。试用信号灯的P、V操作实现这两个进程的互斥。

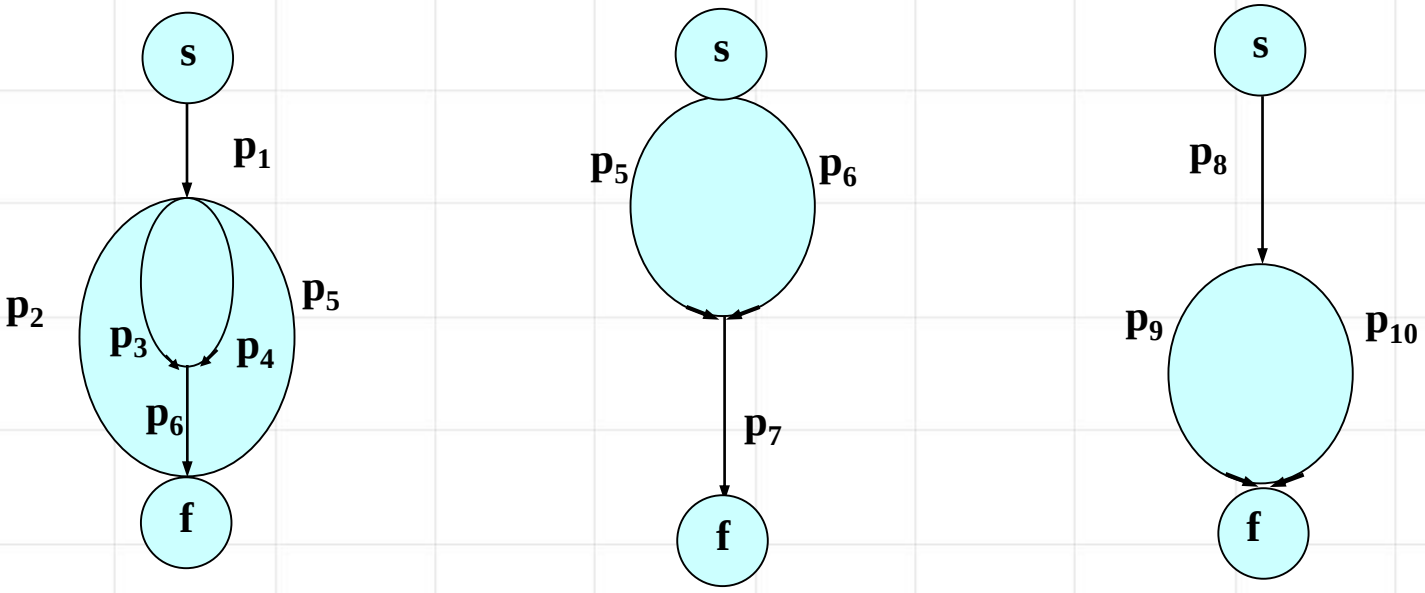
设：mutex为互斥信号灯，初值为1。

```
pa( )                    pb( )
{                        {
      M                        M
   p(mutex);                p(mutex);
    x:=x+1 ;                 x:=x+1 ;
   v(mutex);                v(mutex);
      M                        M
}                        }
```

3. 两类同步问题的解法

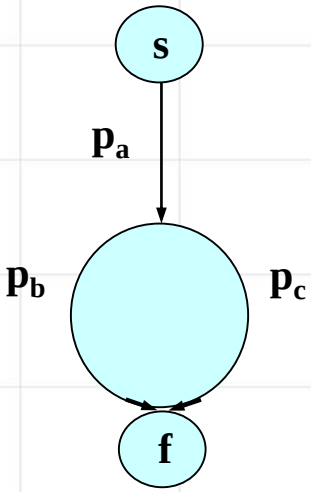
(1) 合作进程的执行次序

① 进程流图



进程流图示例

② 例：Pa、Pb、Pc为一组合作进程，其进程流图如图所示，试用信号灯的p、v操作实现这三个进程的同步。



3个合作进程的进程流图

i 分析任务的同步关系

任务启动后 p_a先执行，当它结束后，p_b、p_c可以 开始执行，p_b、p_c 都执行完毕后，任务终止。

ii 信号灯设置

设两个同步信号灯s_b、s_c分别表示进程p_b和p_c能否开始执行，其初值均为0。

iii 同步描述

p_a	p_b	p_c
M	$p(s_b)$;	$p(s_c)$;
$v(s_b)$;	M	M
$v(s_c)$;	M	M

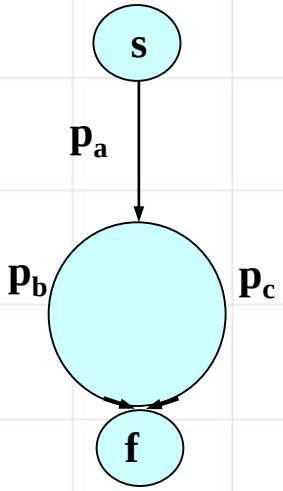
程序 task4

```
main( )
{
   int  sb=0;  /*表示Pb进程能否开始执行*/
   int  sc=0;  /*表示Pc进程能否开始执行*/
   cobegin
      pa( );
      pb( );
      pc( );
   coend
}
```

```
pa( )
{
    M
   v(sb);
   v(sc);
}
```

```
pb( )
{
      p(sb);
       M
       M
}
```

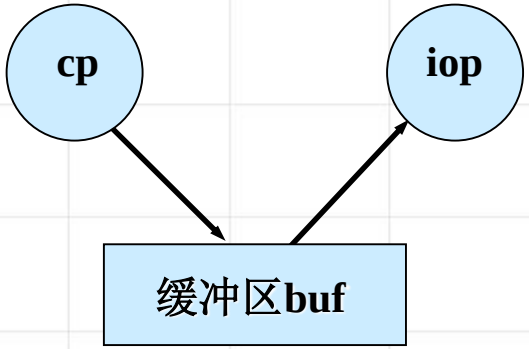
```
pc( )
{
      p(sc);
       M
       M
}
```



3个合作进程的进程流图

(2) 共享缓冲区的合作进程的同步的解法

计算进程 cp和打印进程 iop公用一个单缓冲，为了完成正确的计算与打印，试用信号灯的p、v操作实现这两个进程的同步。



共享缓冲区的合作进程的同步示意图

① 两个进程的任务

计算进程cp经过计算，将计算结果送入buf；

打印进程iop把buf中的数据取出打印。

② 分析任务的同步关系

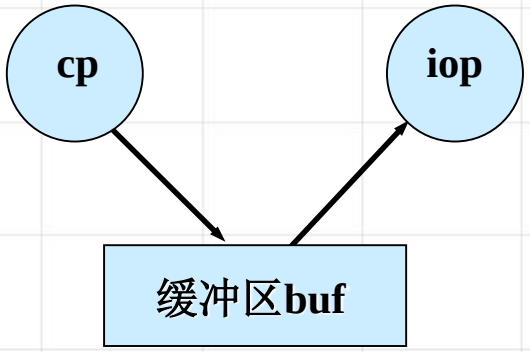
当cp进程把计算结果送入buf时，iop进程才能从buf中取出结果去打印，否则必须等待。

当iop进程把buf中的数据取出打印后，cp进程才能把下一个计算结果数据送入buf中，否则必须等待。

③ 信号灯设置

s_a：表示缓冲区中是否有可供打印的计算结果，其初值为0。

s_b：表示缓冲区有无空位置存放新的信息，其初值为1。



共享缓冲区的合作进程的同步示意图

④ 同步描述

cp:

M

产生一个数据；

$p(s_b)$；

将数据放入buf ；

$v(s_a)$；

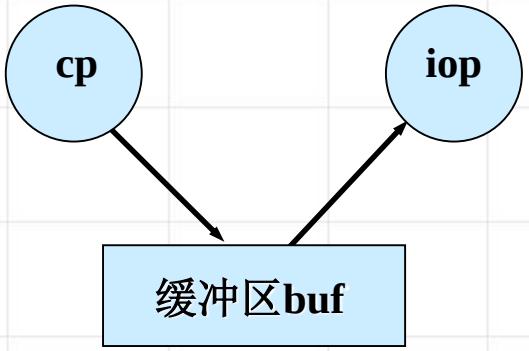
iop:

$p(s_a)$；

从buf中取 数据；

$v(s_b)$；

打印；

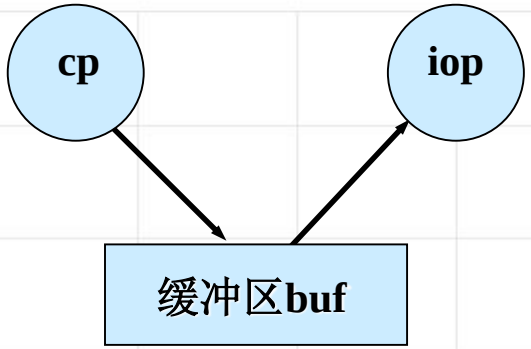


共享缓冲区的合作进程的同步示意图

⑤ 程序描述

```
程序 task5
main( )
{
    int sa=0;        /*<表示buf中有无信息<   */
    int sb=1;        /*<表示buf中有无空位置<*/
    cobegin
        cp( ); iop( );
    coend
}
cp( )
{
    while(计算未完成)
    {
        得到一个计算结果;
        p(sb);
        将数送到缓冲区中;
        v(sa);
    }
}
```

```
iop( )
{
    while(打印工作未完成)
    {
        p(sa);
        从缓冲区中取一数;
        v(sb);
        从打印机上输出;
    }
}
```



共享缓冲区的合作进程的同步示意图