華中科技大學
计算机科学与技术学院
School of Computer Science & Technology, HUST

操作系统原理

第四章 进程及进程管理

廖小飞

提纲

- 进程的引入
- 进程概念
- 进程控制
- 进程的相互制约关系
- 进程同步机构
- 进程互斥与同步的实现
- 线程
- 进程调度

进程及进程管理——进程的引入

1. 顺序程序及特点

(1) 计算

程序的一次执行过程称为一个计算，它由许多简单操作所组成。

(2) 程序的顺序执行

一个计算的若干操作必须按照严格的先后次序顺序地执行，这类计算过程就是程序的顺序执行过程。

(3) 顺序程序的特点

① 单道系统的工作情况

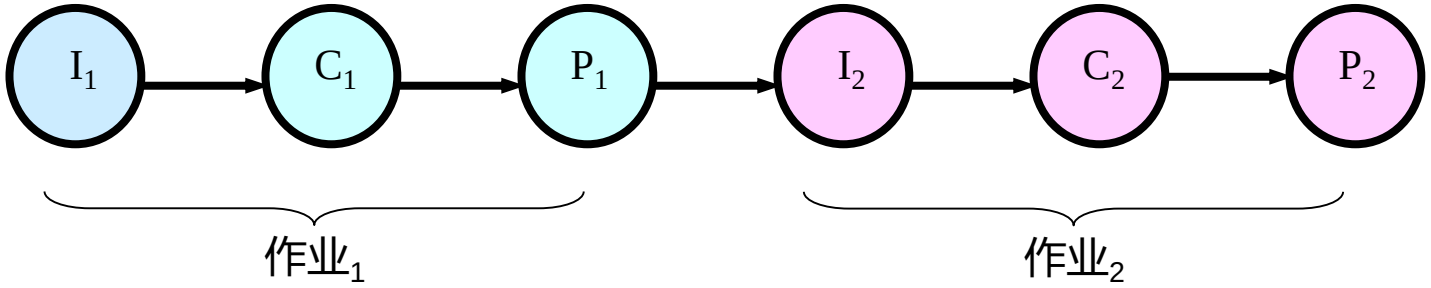
对用户作业的处理 ——

首先输入用户的程序和数据；然后进行计算；最后打印计算结果，即有三个顺序执行的操作。

I：输入操作

C：计算操作

P：输出操作



单用户系统中操作的先后次序图

② 顺序程序的特点

- 顺序性 —— 处理机的操作严格按照程序所规定的顺序执行。
- 封闭性 —— 程序一旦开始执行，其计算结果不受外界因素的影响。
- 可再现性 —— 程序执行的结果与它的执行速度无关（即与时间无关），而只与初始条件有关。

2. 并发程序

(1) 多道系统的工作情况

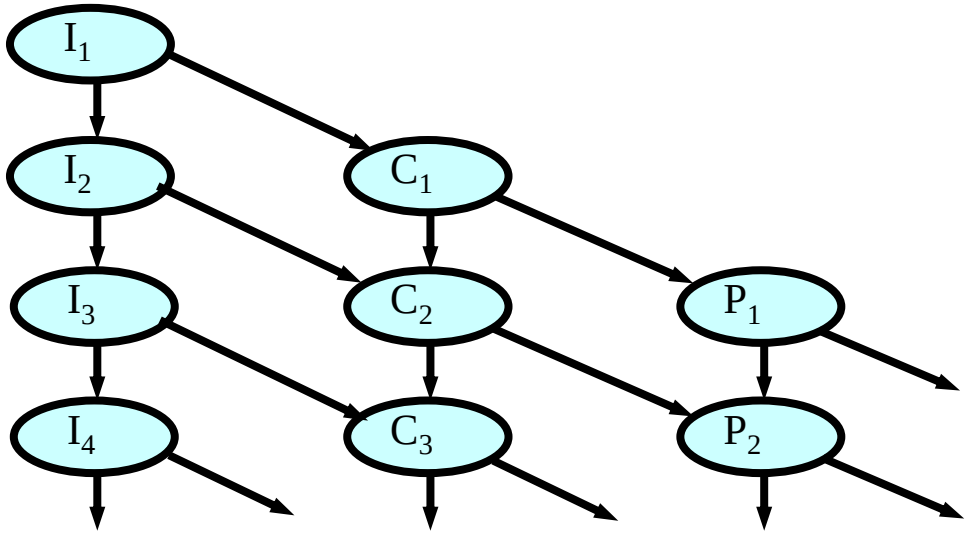
对n个用户作业的处理 ——

作业$_1$: I_1 C_1 P_1

作业$_2$: I_2 C_2 P_2

⋮ ⋮ ⋮ ⋮

作业$_n$: I_n C_n P_n



多用户系统中操作的先后次序图

- 哪些程序段的执行必须是顺序的？为什么？
- 哪些程序段的执行是并行的？为什么？

(2) 什么是程序的并发执行

① 定义

若干个程序段同时在系统中运行，这些程序段的执行在时间上是重叠的，一个程序段的执行尚未结束，另一个程序段的执行已经开始，即使这种重叠是很小的一部分，也称这几个程序段是并发执行的。

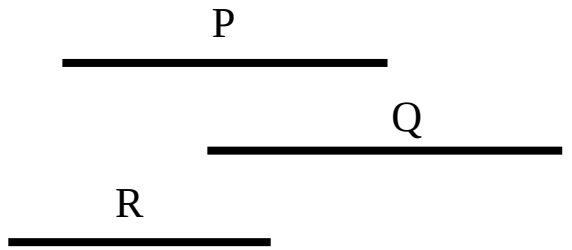
② 三个并发执行的程序段

③ 并行语句记号

cobegin

S1；S2；……；Sn；

coend



三个并发进程

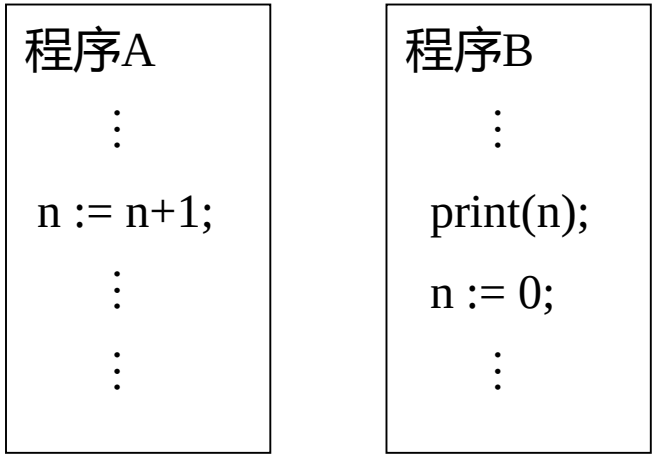
(3) 并发程序的特点

① 失去程序的封闭性和可再现性

若一个程序的执行可以改变另一个程序的变量，那么，后者的输出就可能有赖于各程序执行的相对速度，即失去了程序的封闭性特点。

例如：

讨论共享公共变量的两个程序，执行时可能产生的不同结果。程序A执行时对n做加1的操作；程序B打印出n值，并将它重新置为零。



共享变量的两个程序

进程及进程管理——进程的引入

ii 失去程序的封闭性和可再现性的讨论

程序A	程序B
⋮	⋮
n := n+1;	print(n);
⋮	n := 0;
⋮	⋮

共享变量的两个程序（n的初值为10）

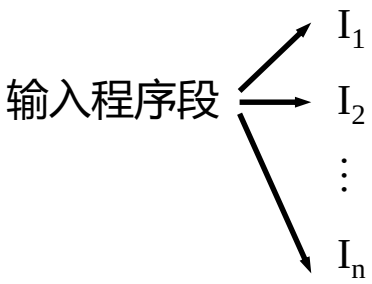
程序A的n :=n+1与程序B的两个语句的关系	之前	之后	之间
n的取值	10	10	10
打印结果	11	10	10
n的最终取值	0	1	0

进程及进程管理——进程的引入

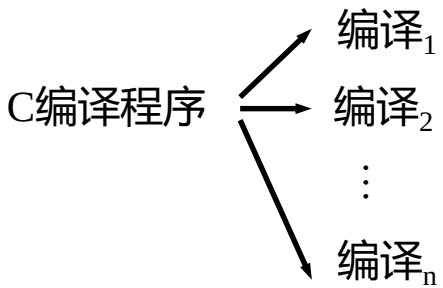
② 程序与计算不再一一对应

一个程序可以对应多个计算。

例1：



例2：



一个程序对应多个计算的例子

③ 程序并发执行的相互制约

- 间接的相互制约关系 —— 资源共享
- 直接的相互制约关系 —— 公共变量

3. 什么是与时间有关的错误

程序并发执行时，若共享了**公共变量**，其执行结果与各并发程序的**相对速度**有关，即给定相同的初始条件，若不加以控制，也可能得到不同的结果，此为**与时间有关的错误。**

提纲

- 进程的引入
- 进程概念
- 进程控制
- 进程的相互制约关系
- 进程同步机构
- 进程互斥与同步的实现
- 线程
- 进程调度

1. 进程定义

运行 ⟶ 暂停 ⟶ 运行

(1) 什么是进程

所谓进程，就是一个程序在给定**活动空间**和**初始环境**下，在一个处理机上的**执行过程**。

(2) 进程与程序的区别

① 程序是**静态**的概念，进程是**动态**的概念；

② 进程是一个独立运行的活动单位；

③ 进程是竞争系统资源的基本单位；

④ 一个程序可以对应多个进程，一个进程至少包含一个程序。

2. 进程的状态

(1) 进程的基本状态

① 运行状态(running)

该进程已获得运行所必需的资源，它的程序正在处理机上执行。

② 等待状态(wait)

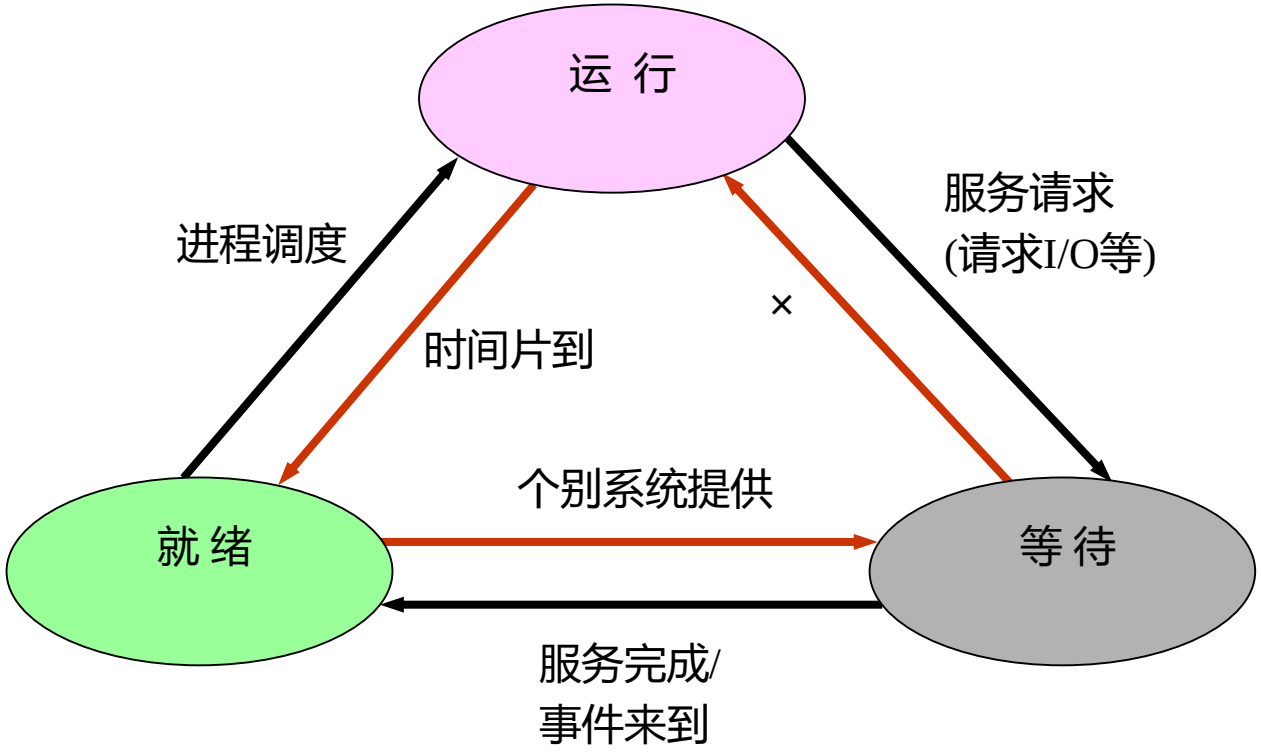
进程正等待着某一事件的发生而暂时停止执行。这时，即使给它CPU控制权，它也无法执行。

③ 就绪状态(ready)

进程已获得除CPU之外的运行所必需的资源，一旦得到CPU控制权，立即可以运行。

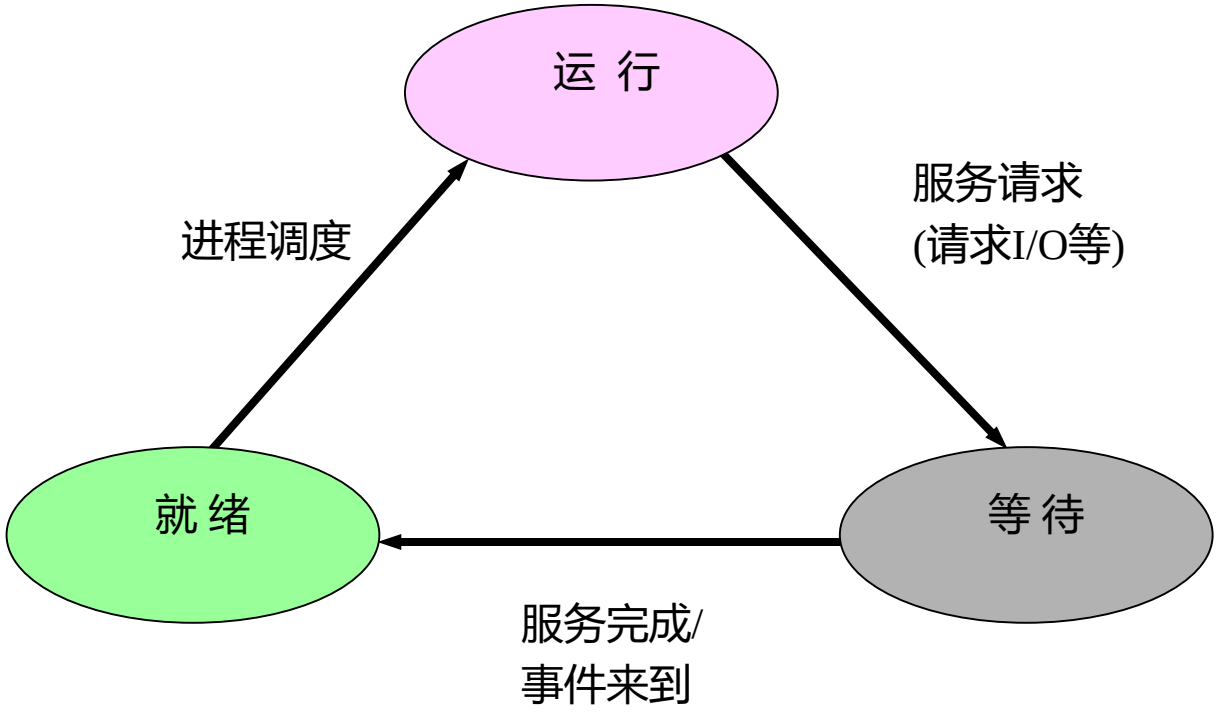
(2) 进程状态的变迁

① 进程状态可能的变迁



进程状态变迁图

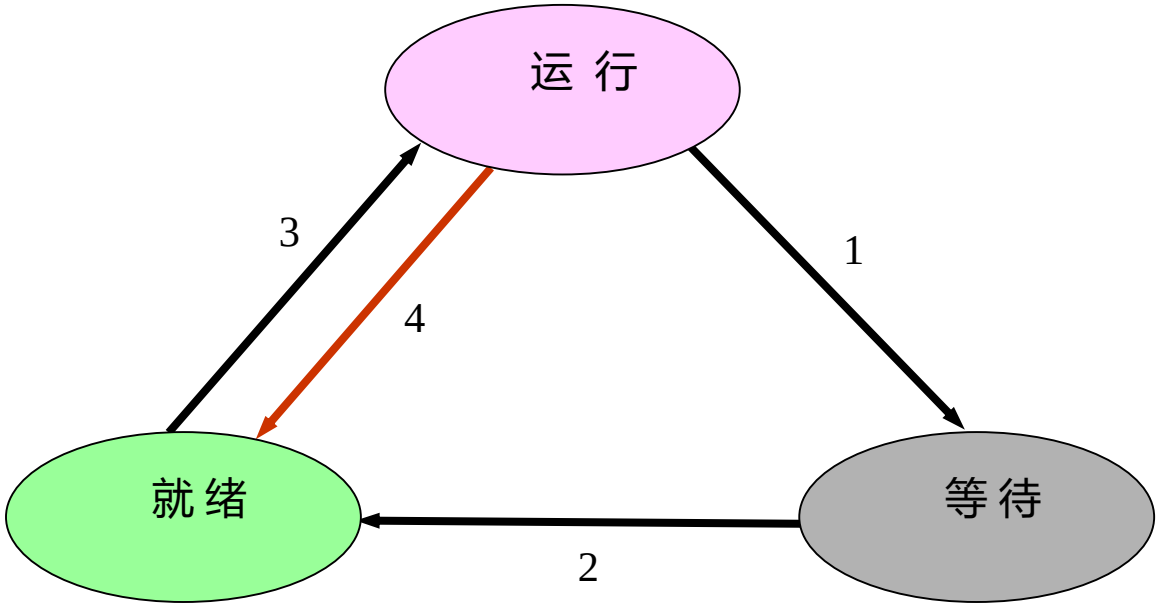
② 具有进程基本状态的变迁图



进程状态变迁图

进程及进程管理——进程概念

③ 讨论进程状态的变迁



进程状态变迁的讨论

变迁1——> 变迁3，是否会发生？需要什么条件?

变迁4——> 变迁3，是否会发生？需要什么条件?

(3) 讨论在多进程操作系统环境下程序的执行

① 例1：讨论3个排序程序在不同的操作系统环境中执行结果

程序A：冒泡排序算法，在屏幕的左1/3处开设窗口显示其排序过程；

程序B：堆排序算法，在屏幕的中1/3处开设窗口显示其排序过程；

程序C：快速排序算法，在屏幕的右1/3处开设窗口显示其排序过程。

讨论在不支持多进程的操作系统下运行和在支持多进程的操作系统下运行的情况

进程及进程管理——进程概念

i **在不支持多进程的操作系统下运行**

依次运行程序A、程序B、程序C。

ii 在支持多进程的操作系统下运行

- 建立进程A、B、C；对应的程序分别是程序A、B、C；
- 若系统采用时间片轮转的调度策略，则在屏幕上有3个窗口，同时显示3个排序过程。

实际上这3个程序在轮流地占用CPU时间，由于CPU的高速度，使我们看到的是这3个程序在同时执行。

② 例2：讨论2个程序在不同的操作系统环境中执行结果

程序C：打印工资报表的程序；

程序D：计算1000以内所有素数并显示最后结果。

讨论在不支持多进程的操作系统下运行和在支持多进程的操作系统下运行。

进程及进程管理——进程概念

i 在不支持多进程的操作系统下运行

依次运行程序C、程序D，可以看到，先是打印机不停地打印工资报表，打完后，接着运行程序C，不停地计算，最后显示所计算的结果。

ii 在支持多进程的操作系统下运行

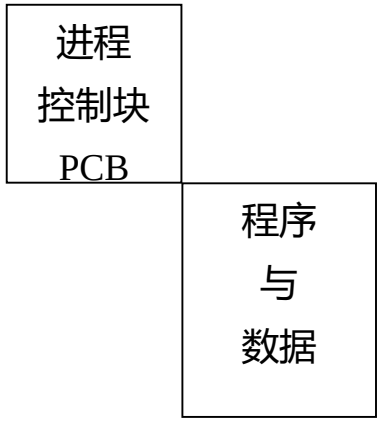
- 建立进程C、D；对应的程序分别是程序C、D；
- 由于进程C是I/O量较大的进程，而进程D是计算量较大的进程，故在系统进程调度的控　　制下，两个进程并发执行。可以看到打印机不断打印工资报表；而处理机不停地计算，最后屏幕显示计算的结果。

3. 进程描述

(1) 什么是进程控制块

描述进程与其他进程、系统资源的关系以及进程在各个不同时期所处的状态的数据结构，称为进程控制块PCB (process control block)。

(2) 进程的组成



进程组成的示意图

① 程序与数据

描述进程本身所应完成的功能

② PCB

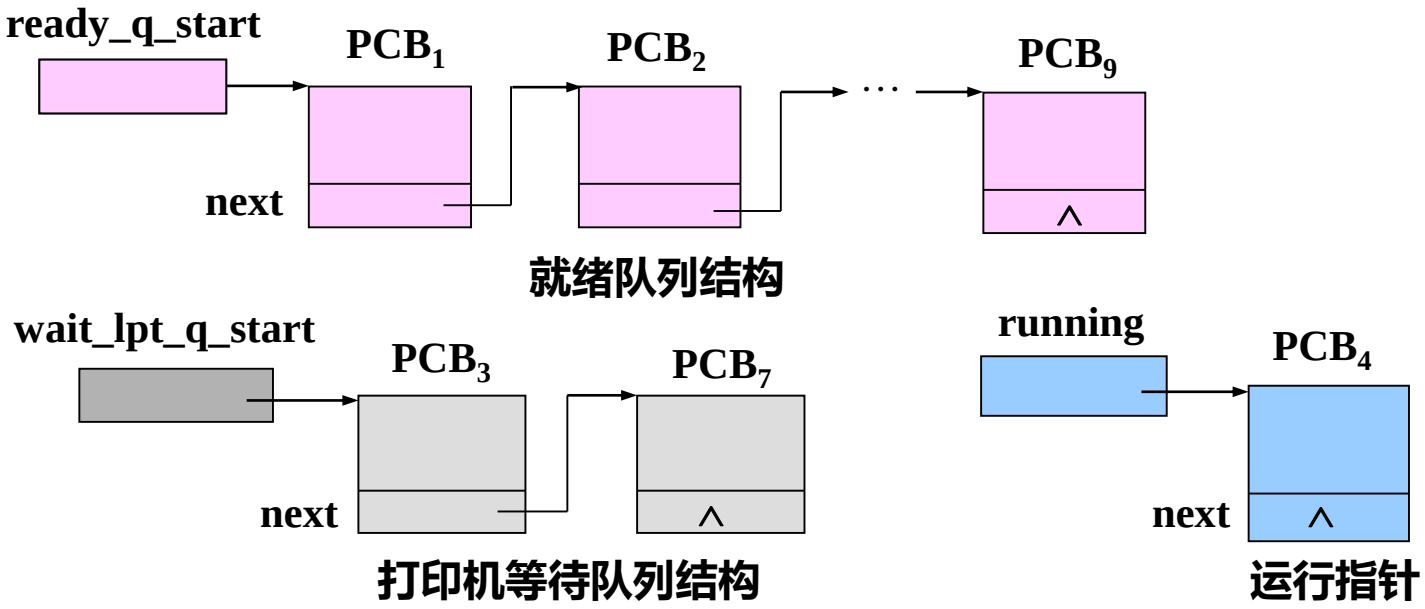
进程的动态特征，该进程与其他进程和系统资源的关系。

(3) 进程控制块的主要内容

① **进程标识符**　　进程符号名或内部 id号

② **进程当前状态**　　本进程目前处于何种状态

大量的进程如何组织?



进程队列结构示例

进程及进程管理——进程概念

③ 当前队列指针next

该项登记了处于同一状态的下一个进程的 PCB地址。

④ 进程优先级

反映了进程要求CPU的紧迫程度。

⑤ CPU现场保护区

当进程由于某种原因释放处理机时，CPU现场信息被保存在PCB的该区域中。

⑥ 通信信息

进程间进行通信时所记录的有关信息。

⑦ 家族联系

指明本进程与家族的联系

⑧ 占有资源清单

提纲

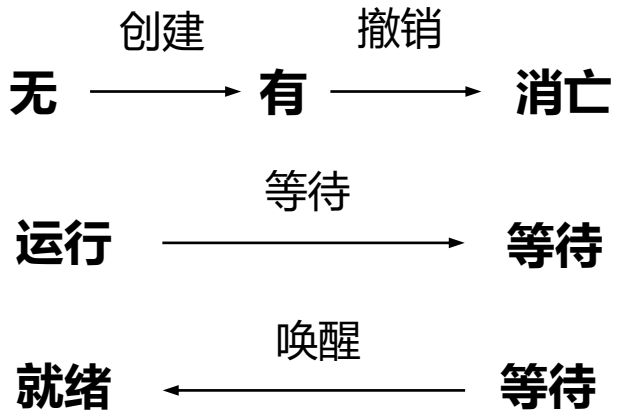
- 进程的引入
- 进程概念
- 进程控制
- 进程的相互制约关系
- 进程同步机构
- 进程互斥与同步的实现
- 线程
- 进程调度

1. 进程控制的概念

(1) 进程控制的职责

对系统中的进程实施有效的管理，负责进程状态的改变。

① 进程状态变化



② 常用的进程控制原语

创建原语、撤消原语、阻塞原语、唤醒原语

(2) 进程创建

① 进程创建原语的形式

create (name， priority)

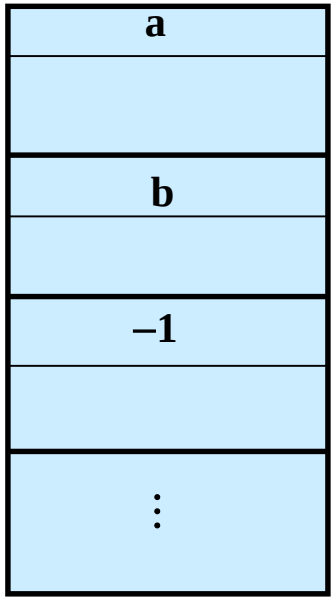
- name为被创建进程的标识符
- priority为进程优先级

② 进程创建原语的功能

创建一个具有指定标识符的进程，建立进程的PCB结构。

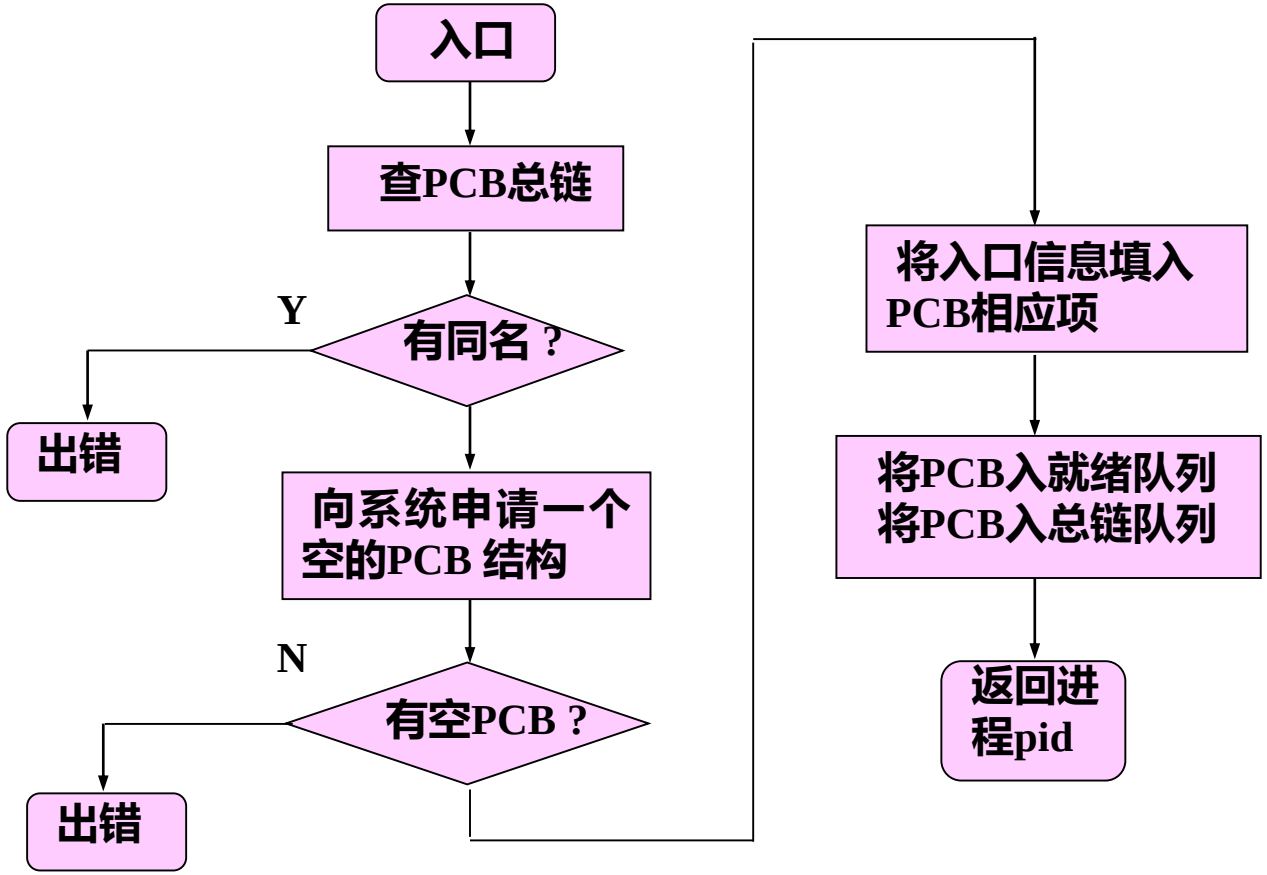
③ 进程创建原语的实现

◆PCB池



PCB池示意图

◆进程创建原语的实现框图



进程创建原语流程图

(3) 进程撤销

① 进程撤销原语的形式

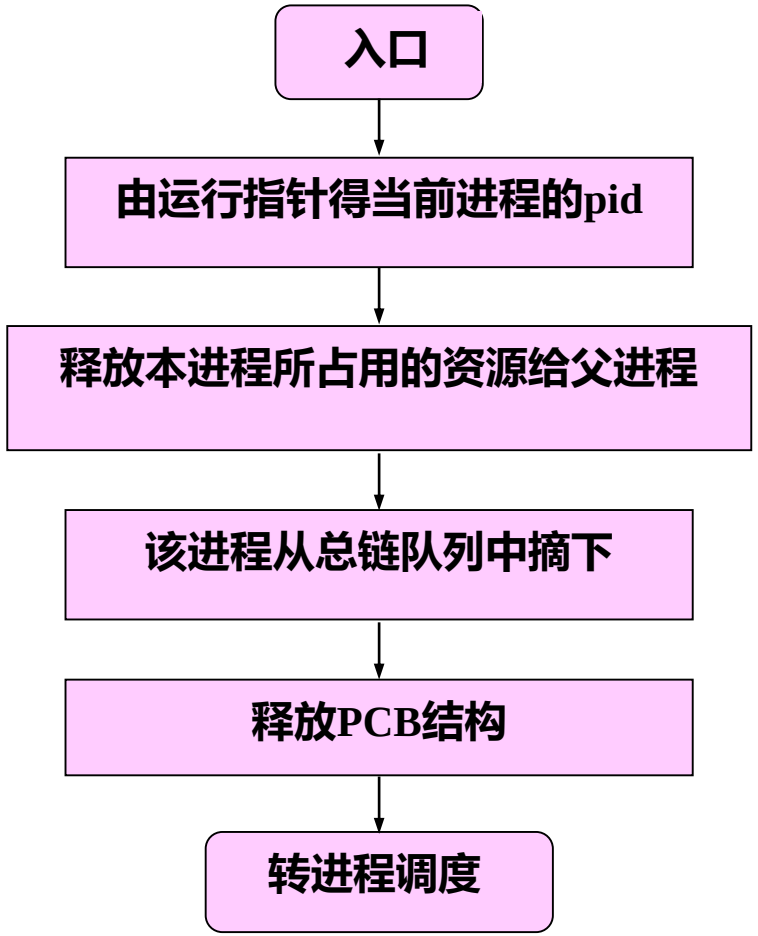
当进程完成任务后希望终止自己时使用进程撤消原语。

kill (或exit)

② 进程撤销原语的功能

撤消当前运行的进程。将该进程的PCB结构归还到PCB资源池，所占用的资源归还给父进程，从总链队列中摘除它，然后转进程调度程序。

③ 进程撤销原语的实现



进程撤销原语流程图

(4) 进程等待

① 进程等待原语的形式

当进程需要等待某一事件完成时，它可以调用等待原语挂起自己。

susp(chan)

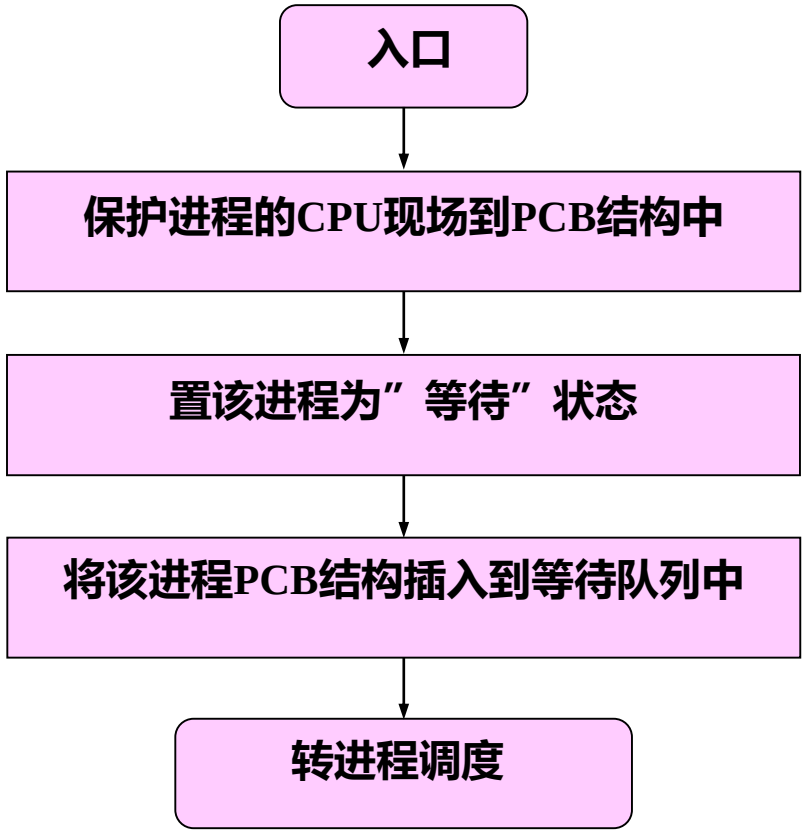
入口参数chan：进程等待的原因

② 进程等待原语的功能

中止调用进程的执行，并加入到等待chan的等待队列中；

最后使控制转向进程调度。

③ 进程等待原语的实现



进程等待原语流程图

(5) 进程唤醒

① 进程唤醒原语的形式

当处于等待状态的进程所期待的事件来到时，由发现者进程使用唤醒原语叫唤醒它。

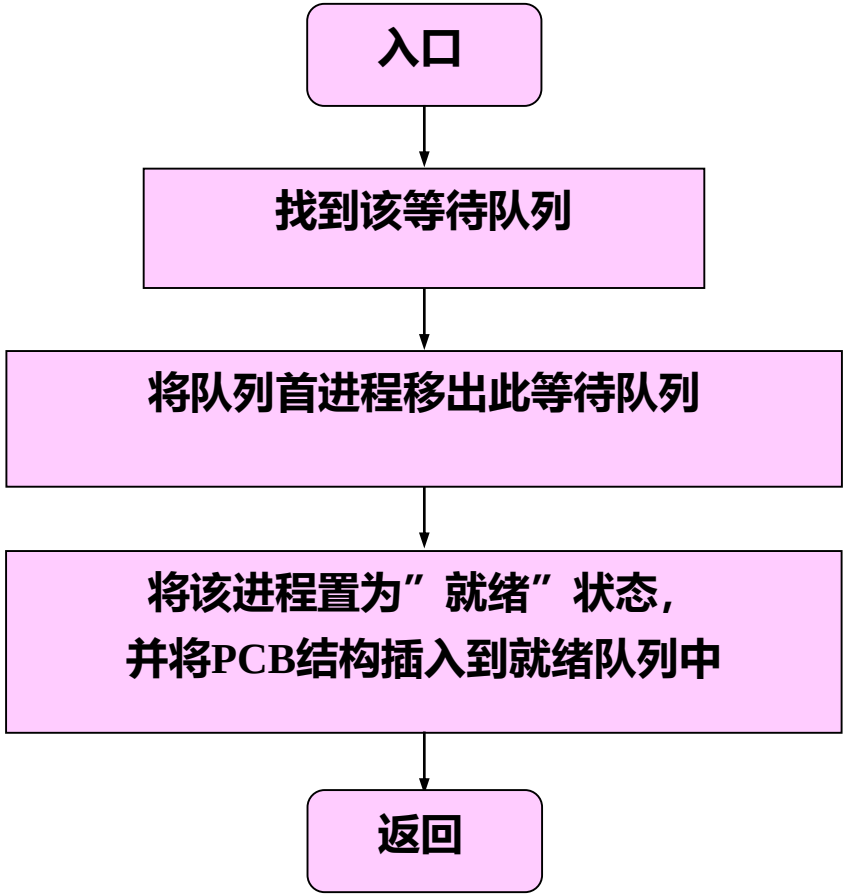
wakeup(chan)

入口参数chan：进程等待的原因。

② 进程唤醒原语的功能

当进程等待的事件发生时，唤醒等待该事件的进程。

③ 进程唤醒原语的实现



进程唤醒原语流程图

提纲

- 进程的引入
- 进程概念
- 进程控制
- 进程的相互制约关系
- 进程同步机构
- 进程互斥与同步的实现
- 线程
- 进程调度

1. 进程互斥的概念

(1) 临界资源

① 例1：两个进程A、B共享一台打印机

设：x代表某航班机座号，p_1和p_2两个售票进程，售票工作是对变量x加1。这两个进程在一个处理机C上并发执行，分别具有内部寄存器r_1和r_2。

② 例2：两个进程共享一个变量x

两个进程共享一个变量x时，两种可能的执行次序：

- A：

p_1： $r_1 := x$； $r_1 := r_1+1$； $x := r_1$ ；

p_2： $r_2 := x$； $r_2 := r_2+1$； $x := r_2$ ；

- B：

p_1： $r_1 := x$； $r_1 := r_1+1$； $x := r_1$ ；

p_2： $r_2 := x$； $r_2 := r_2+1$； $x := r_2$ ；

- 设x的初值为10，两种情况下的执行结果：

情况A： x = 10+2

情况B： x = 10+1

③ 临界资源的定义

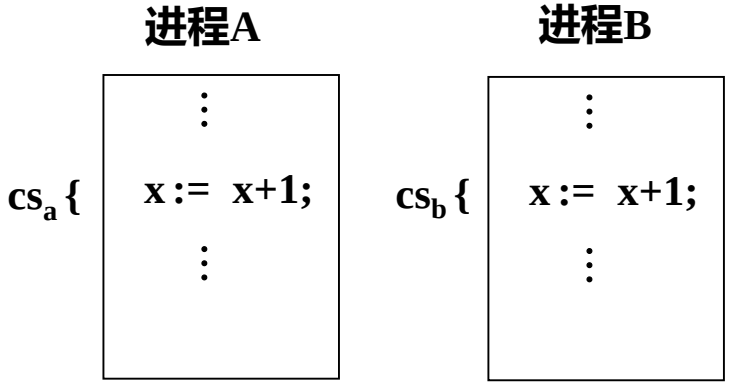
一次仅允许一个进程使用的资源称为**临界资源**。

硬件：如输入机、打印机、磁带机等

软件：如公用变量、数据、表格、队列等

(2) 临界区

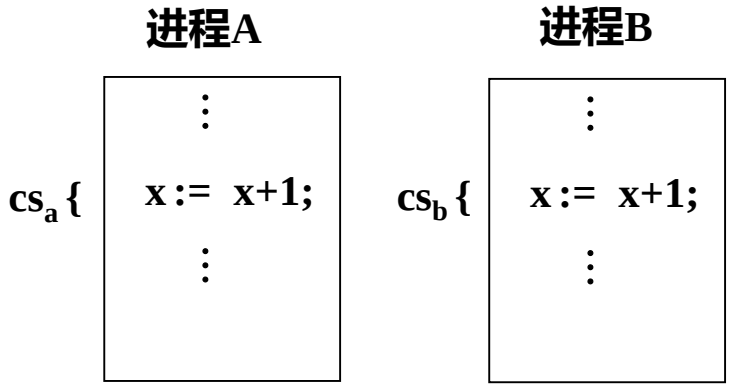
临界区是进程中对公共变量 (或存储区)进行审查与修改的**程序段**，称为相对于该公共变量的临界区。



进程临界区示意图

(3) 互斥

在操作系统中，当某一进程正在访问某一存储区域时，就不允许其他进程来读出或者修改存储区的内容，否则，就会发生后果无法估计的错误。进程间的这种**相互制约关系**称为**互斥**。



进程临界区示意图

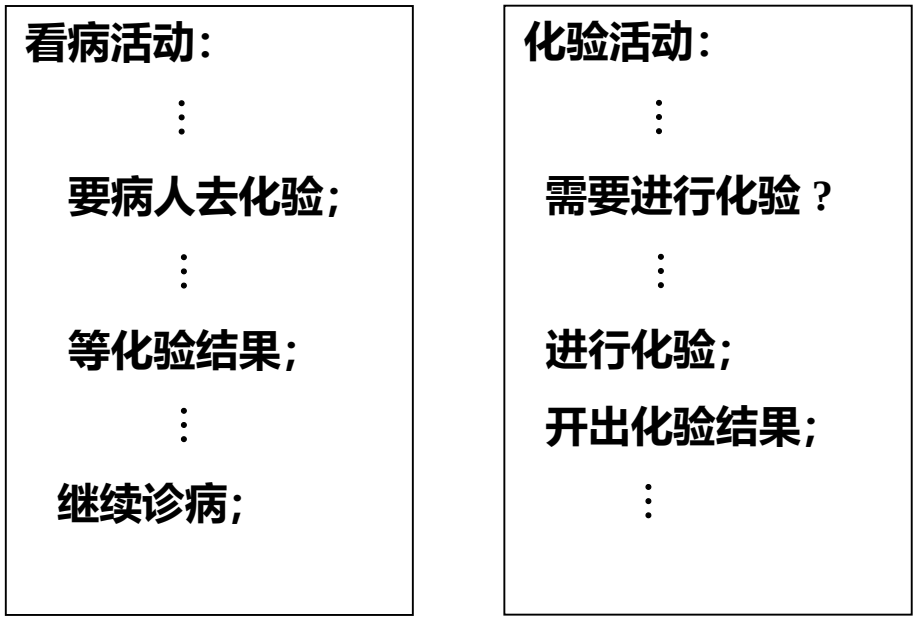
2. 进程同步的概念

(1) 什么是进程同步

并发进程在一些**关键点**上可能需要**互相等待**与**互通消息**，这种相互制约的等待与互通消息称为**进程同步**。

(2) 进程同步例子

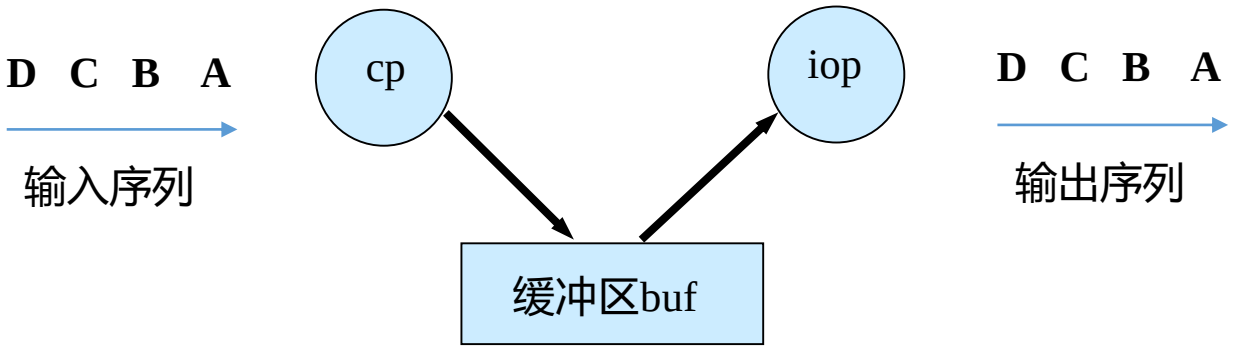
① 病员就诊



进程同步活动示意图

② 共享缓冲区的计算进程与打印进程的同步

计算进程 cp和打印进程 iop公用一个单缓冲



两个进程共享一个缓冲区示意图

提纲

- 进程的引入
- 进程概念
- 进程控制
- 进程的相互制约关系
- **进程同步机构**
- 进程互斥与同步的实现
- 线程
- 进程调度

1. 锁和上锁、开锁操作

(1) 什么是锁

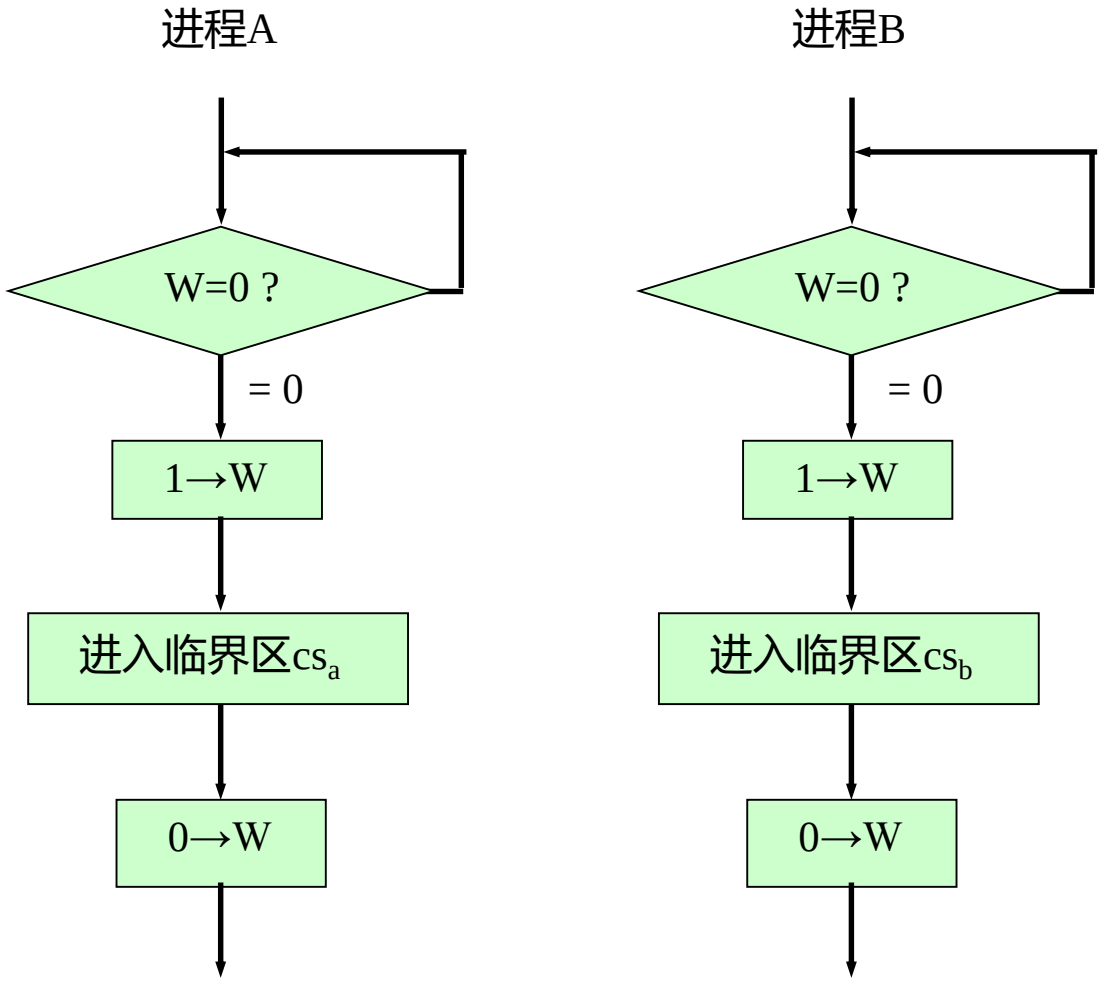
用变量w代表某种资源的状态，w称为“锁” 。

(2) 上锁操作和开锁操作

- 检测w的值 (是0还是1)；
- 如果w的值为1，继续检测；
- 如果w的值为0，将锁位置1 (表示占用资源)，进入临界区执行。**(此为上锁操作)**

- 临界资源使用完毕，将锁位置0。 **(此为开锁操作)**

(3) 进程使用临界资源的操作



两个进程使用临界资源的操作

(4) 上锁原语和开锁原语

① 上锁原语

```
算法  lock
输入：锁变量w
输出：无
{
    test：   if (w为1)
    goto  test;
                /* 测试锁位的值   */
    else   w=1;       /*   上锁   */
}
```

② 开锁原语

```
算法  unlock
    输入：锁变量w
    输出：无
    {
        w=0；/*开锁*/
    }
```

2. 信号灯和P、V操作

(1) 什么是信号灯

信号灯是一个确定的二元组 (s，q)，s是一个具有非负初值的整型变量，q是一个初始状态为空的队列。操作系统利用信号灯的状态对并发进程和共享资源进行控制和管理。

信号灯是整型变量。

变量值 ≥ 0 时，表示绿灯，进程执行；

变量值 ＜ 0 时，表示红灯，进程停止执行。

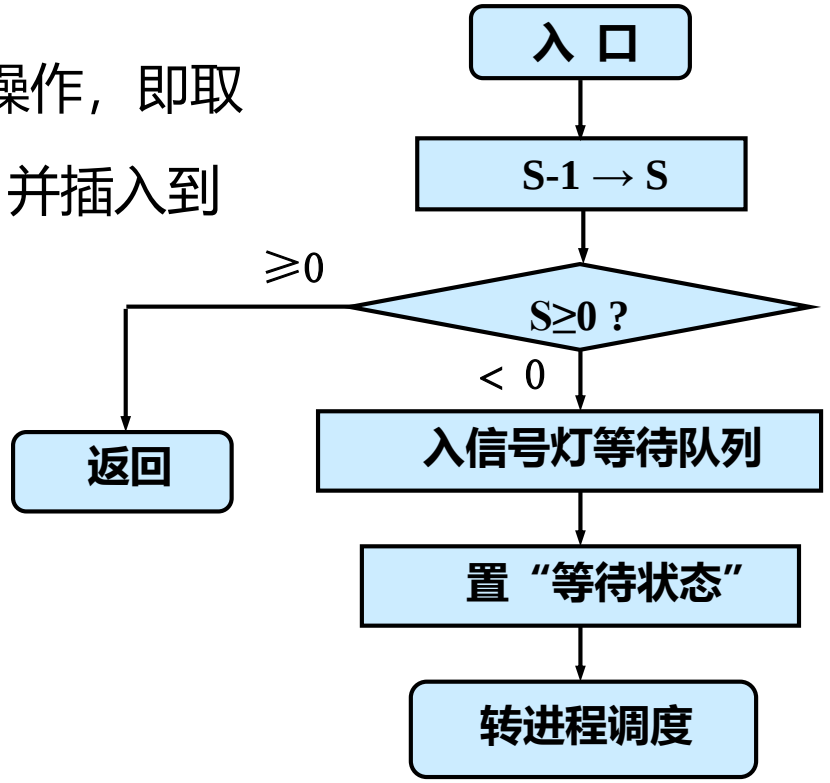
注意：创建信号灯时，应准确说明信号灯 s 的意义和初值(这个初值绝不能为负值)。

(2) P 操作

① P 操作的定义

对信号灯s的 p操作记为 p(s)。p(s)是一个不可分割的原语操作，即取信号灯值减1，若相减结果为负，则调用p(s)的进程被阻，并插入到该信号灯的等待队列中，否则可以继续执行。

② P 操作的实现



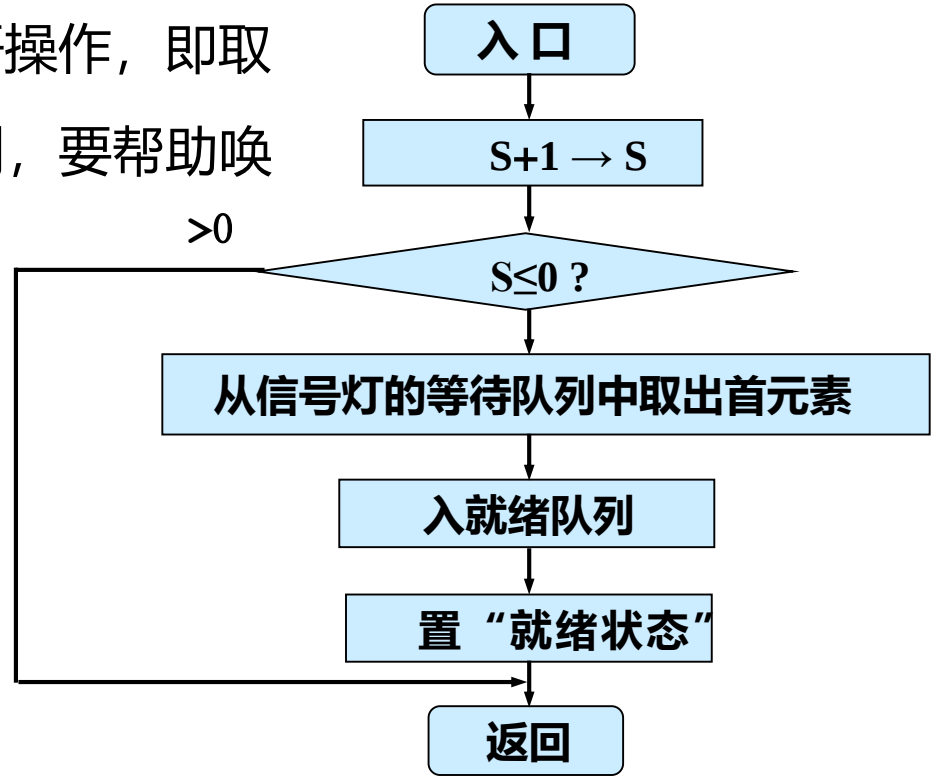
P 操作原语流程图

(3) V 操作

① V 操作的定义

对信号灯s的 v操作记为 v(s)。v(s)是一个不可分割的原语操作，即取信号灯值加1，若相加结果大于零，进程继续执行，否则，要帮助唤醒在信号灯等待队列上的一个进程。

② V 操作的实现



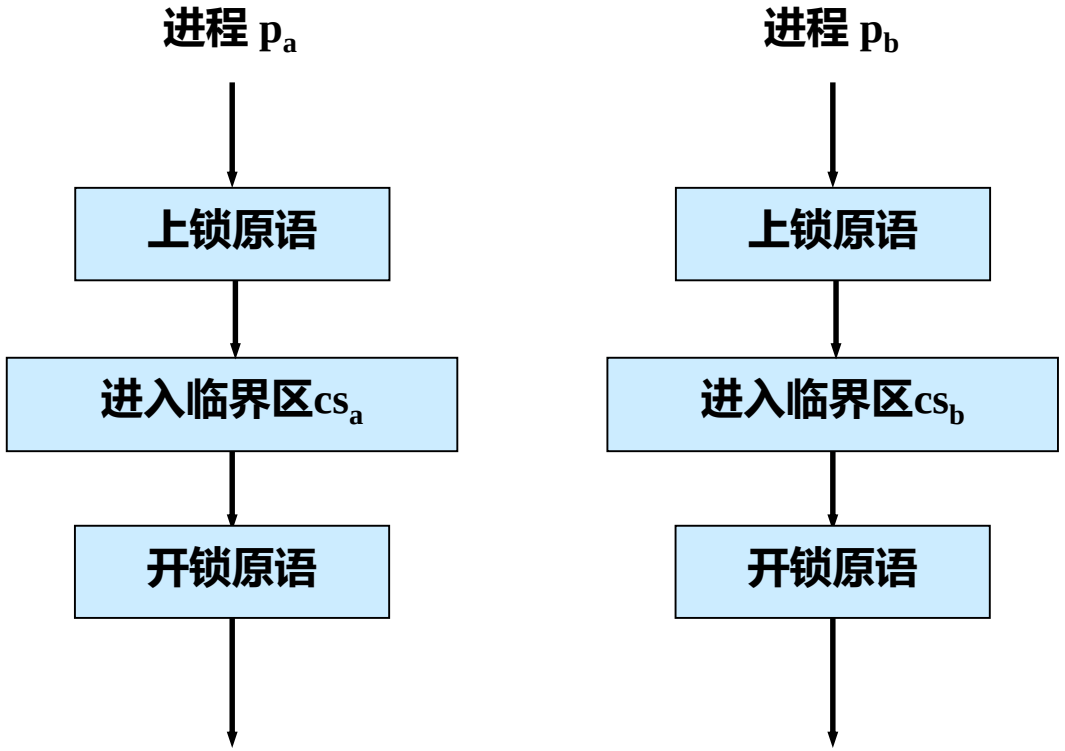
V 操作原语流程图

提纲

- 进程的引入
- 进程概念
- 进程控制
- 进程的相互制约关系
- 进程同步机构
- 进程互斥与同步的实现
- 线程
- 进程调度

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

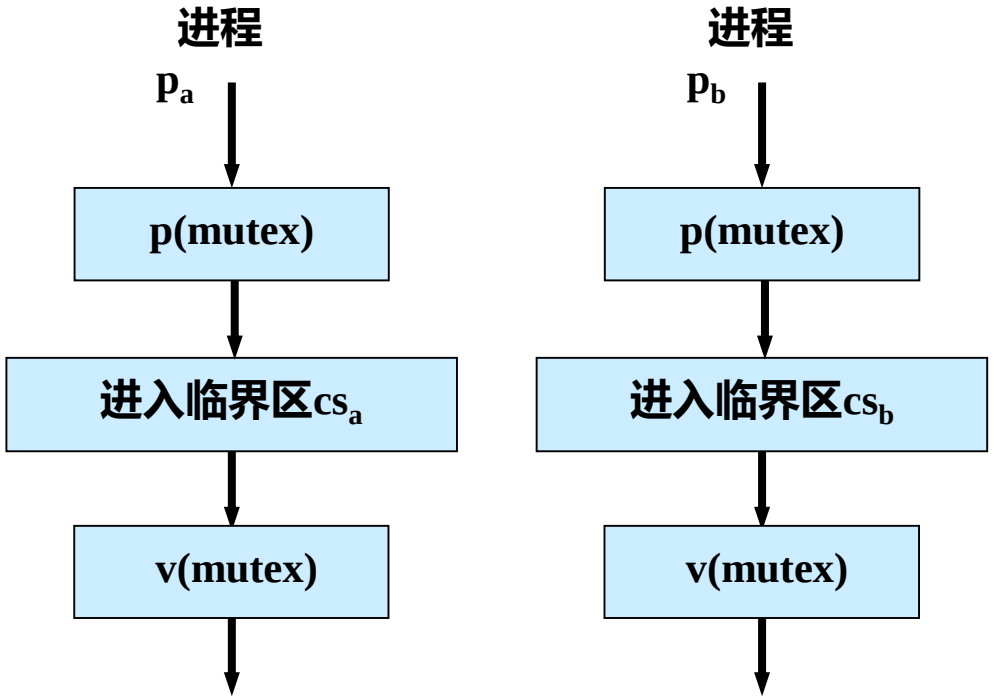
```
main( )
{
   int w=1;        /* 互斥锁 */
   cobegin
     pa( );
     pb( );
   coend
}
```

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

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

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( )
{                              {
        ⋮                              ⋮
  p(mutex);                      p(mutex);
      csa ;                          csb ;
   v(mutex);                      v(mutex);
        ⋮                              ⋮
}                              }
```

(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。

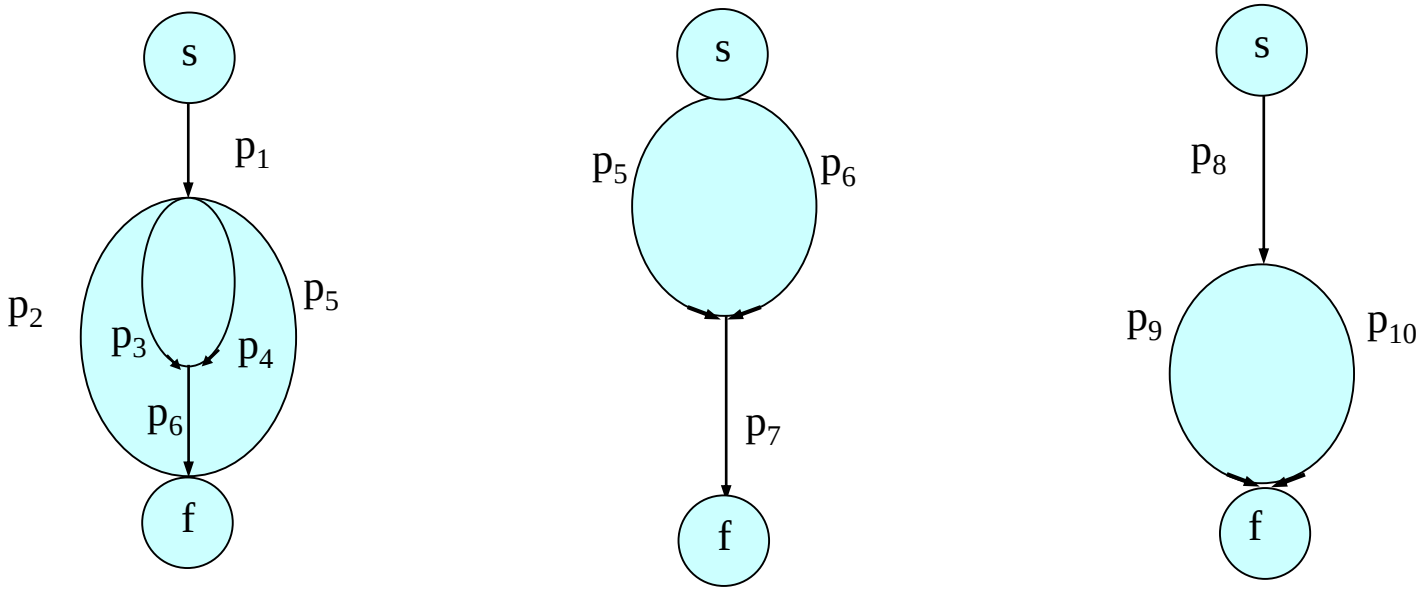
设：mutex为互斥信号灯，初值为1。

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

3. 两类同步问题的解法

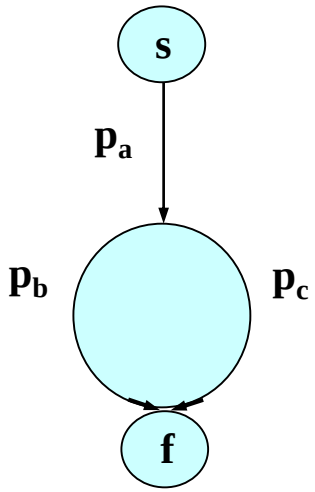
(1) 合作进程的执行次序

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**

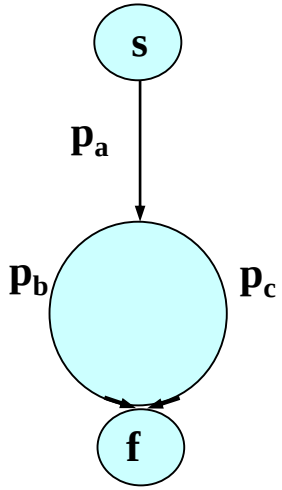
p_a	p_b	p_c
⋮	$p(s_b)$;	$p(s_c)$;
$v(s_b)$;	⋮	⋮
$v(s_c)$;	⋮	⋮

进程及进程管理——进程互斥与同步的实现

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
}
```

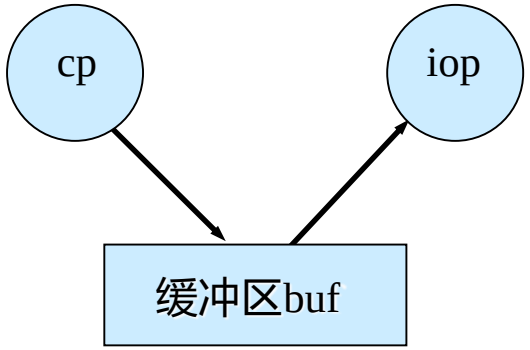
```
pa( )                pb( )                pc( )
{                    {                    {
    ⋮                   p(sb);               p(sc);
  v(sb);                ⋮                    ⋮
  v(sc);                ⋮                    ⋮
}                    }                    }
```



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

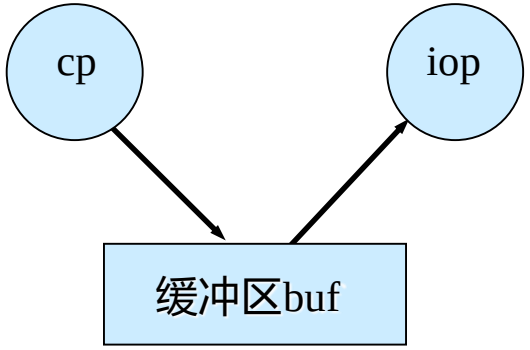
当cp进程把计算结果送入buf时，iop进程才能从buf中取出结果去打印，否则必须等待。

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

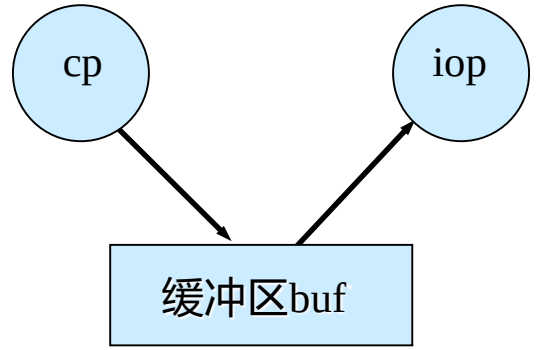
④ 同步描述

cp:

```
⋮
产生一个数据;
p(sb);
将数据放入buf ;
v(sa);
```

iop:

```
p(sa);
从buf中取 数据;
v(sb);
打印;
```

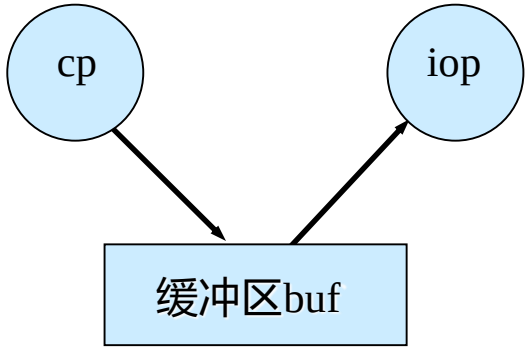


共享缓冲区的合作进程的同步示意图

⑤ 程序描述

进程及进程管理——进程互斥与同步的实现

```
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
}
```



共享缓冲区的合作进程
的同步示意图

```
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;
    }
}
```

4. 生产者——消费者问题

(1) 生产者——消费者问题的例子

① 计算进程和打印进程

计算进程 cp不断产生数据，是生产者；

打印进程 iop不断打印数据，是消费者。

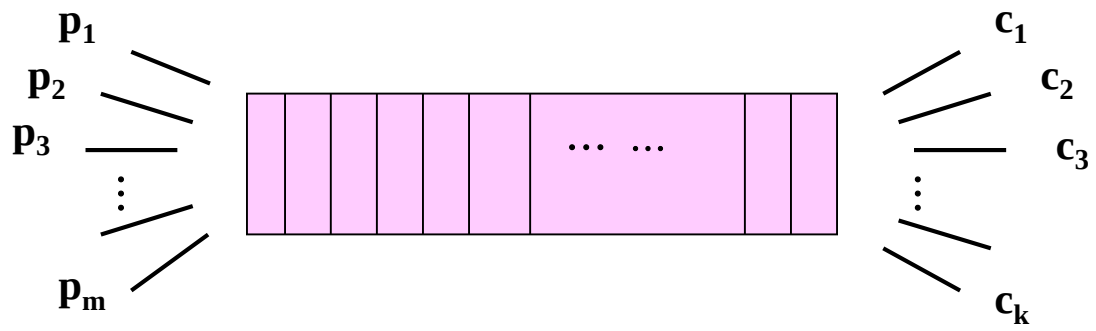
② 通信问题

发消息进程 send不断产生消息，是生产者；

收消息进程 receive不断接收消息，是消费者。

(2) 生产者——消费者问题的一般解答

① 生产者——消费者问题图示



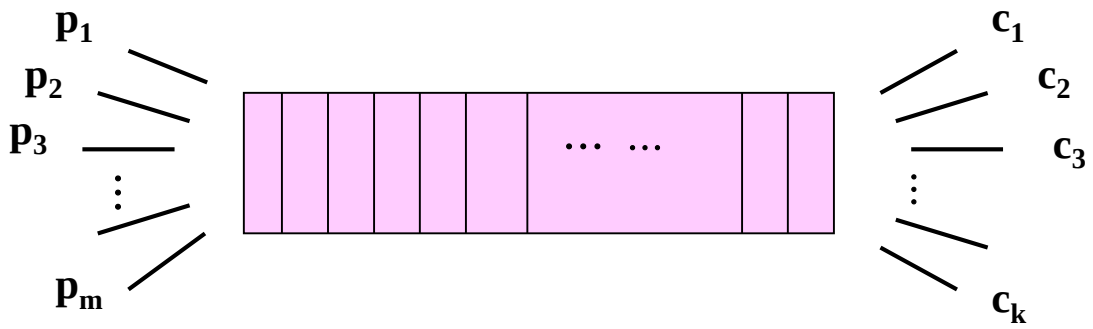
② 生产者与消费者的同步关系

生产者：当有界缓冲区中无空位置时，要等待；

向有界缓冲区放入物品后，要发消息。

消费者：当有界缓冲区中无物品时，要等待；

从有界缓冲区取出物品后，要发消息。



生产者——消费者问题示意图

③ 信号灯设置

i **两个同步信号灯——**

s_b：表示空缓冲区的数目，初值 = n

s_a： 表示满缓冲区 (即信息)的数目，初值 = 0

ii **一个互斥信号灯——**

mutex：表示有界缓冲区是否被占用，初值 = 1

④ 同步描述

生产者:

⋮

$p(s_b)$;

p(mutex);

将数据放入有界缓冲区;

v(mutex);

$v(s_a)$;

消费者:

$p(s_a)$

p(mutex);

从有界缓冲区中取数据;

v(mutex);

$v(s_b)$;

消费;

⑤ 程序描述

```
程序 prod_cons
main( )
{
    int  s_a=0;        /* 满缓冲区的数目 */
    int  s_b=n;        /* 空缓冲区的数目 */
    int  mutex=1;  /* 对有界缓冲区进行操作的互斥信号灯 */
    cobegin
        p_1 ( );  p_2 ( );  … p_m ( );
        c_1 ( );  c_2 ( );  … c_k ( );
    coend
}
```

进程及进程管理——进程互斥与同步的实现


```
pi( )
{
    while(生产未完成)
    {
        ⋮
        生产一个产品;
        p(sb);
        p(mutex);
        送一个产品到有界缓冲
        v(mutex);
        v(sa);
    }
}
```

```
cj( )
{
    while(还要继续消费)
    {
        p(sa);
        p(mutex);
        从有界缓冲区中取产品;
        v(empty);
        v(sb);
        消费一个产品;
        ⋮
    }
}
```

5. 读者——写者问题

① 问题定义

某数据区域（如文件、内存块或寄存器）在多个进程间共享；其中部分进程对该数据区进行只读操作，另一部分进程对数据区进行只写操作。约束如下：

（1）可以有任意多个读者同时读取数据区中的内容；

（2）一次只有一个写者允许向数据区中写；

（3）写者在向数据区中写时，不允许读者同时读取。

(1) 读者优先

程序描述

```
main( )
{
      int  readcount;       /* 读者计数 */
      semaphore x = 1,wsem = 1;
      cobegin
            reader ( );
            writer ( );
      coend
}
```

进程及进程管理——进程互斥与同步的实现


```
reader (){
    while (true){
        P (x);
        readcount++;
        if ( readcount == 1 )
                P(wsem);
        V (x);
        READUNIT();
        P(x);
        readcount--;
        if( readcount == 0 )
                V(wsem);
        V(x);
    }
}
```

```
writer (){
    while (true){
        P (wsem);
        WRITEUNIT();
        V(wsem);
    }
}
```

思考：该解决方案存在何种问题？

(2) 写者优先

程序描述?

1. 进程通信的概念

进程通信是指进程之间直接以较高的效率传递较多数据的信息交互方式。

2. 进程通信方式

(1) 消息缓存通信

- 在消息通信中，接收方和发送方之间有明确的协议和消息格式 。
- 消息缓冲通信方式包括消息缓冲、发送原语和接收原语。

(2) 信箱通信

- 在信箱通信中，需要定义信箱结构，还包括消息发送和接收功能模块，提供发送原语和接收原语。
- 信箱通信中，所使用的信箱可以位于用户空间中，是接收进程地址空间的一部分；也可以放置在操作系统的空间中。

提纲

- 进程的引入
- 进程概念
- 进程控制
- 进程的相互制约关系
- 进程同步机构
- 进程互斥与同步的实现
- **线程**
- 进程调度

1. 什么是线程

(1) 线程定义

线程是比进程更小的活动单位，它是进程中的一个执行路径。

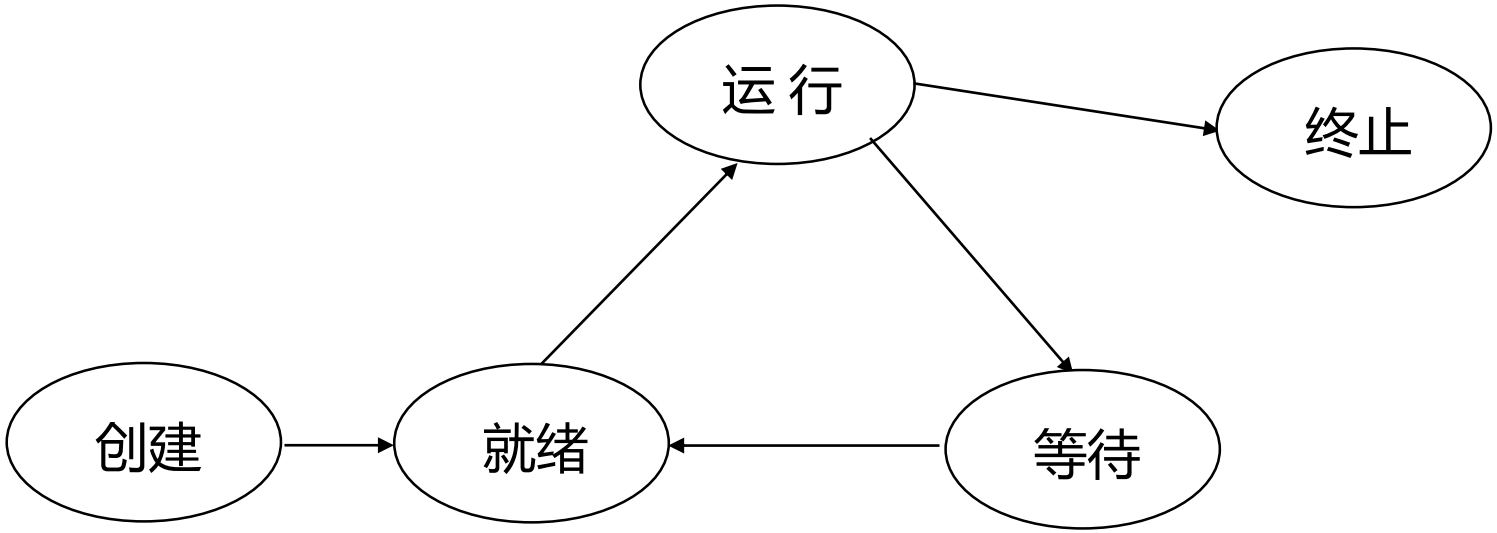
(2) 线程可以这样来描述

- 进程中的一条执行路径；
- 它有自己私用的堆栈和处理机执行环境 ；
- 它与父进程共享分配给父进程的主存；
- 它是单个进程所创建的许多个同时存在的线程中的一个。

2. 线程的特点

- 线程是比进程更小的活动单位，它是进程中的一个执行路径。 创建一个线程比创建一个进程开销要小得多。
- 实现线程间通信十分方便，因为一个进程创建的多个线程可以共享地址区域和数据。
- 线程是一个动态的概念。
- 在进程内创建多线程，可以提高系统的并行处理能力，加快进程的处理速度。。

3. 线程的状态变迁



线程的状态变迁图

提纲

- 进程的引入
- 进程概念
- 进程控制
- 进程的相互制约关系
- 进程同步机构
- 进程互斥与同步的实现
- 线程
- 进程调度

1. 调度/分派结构

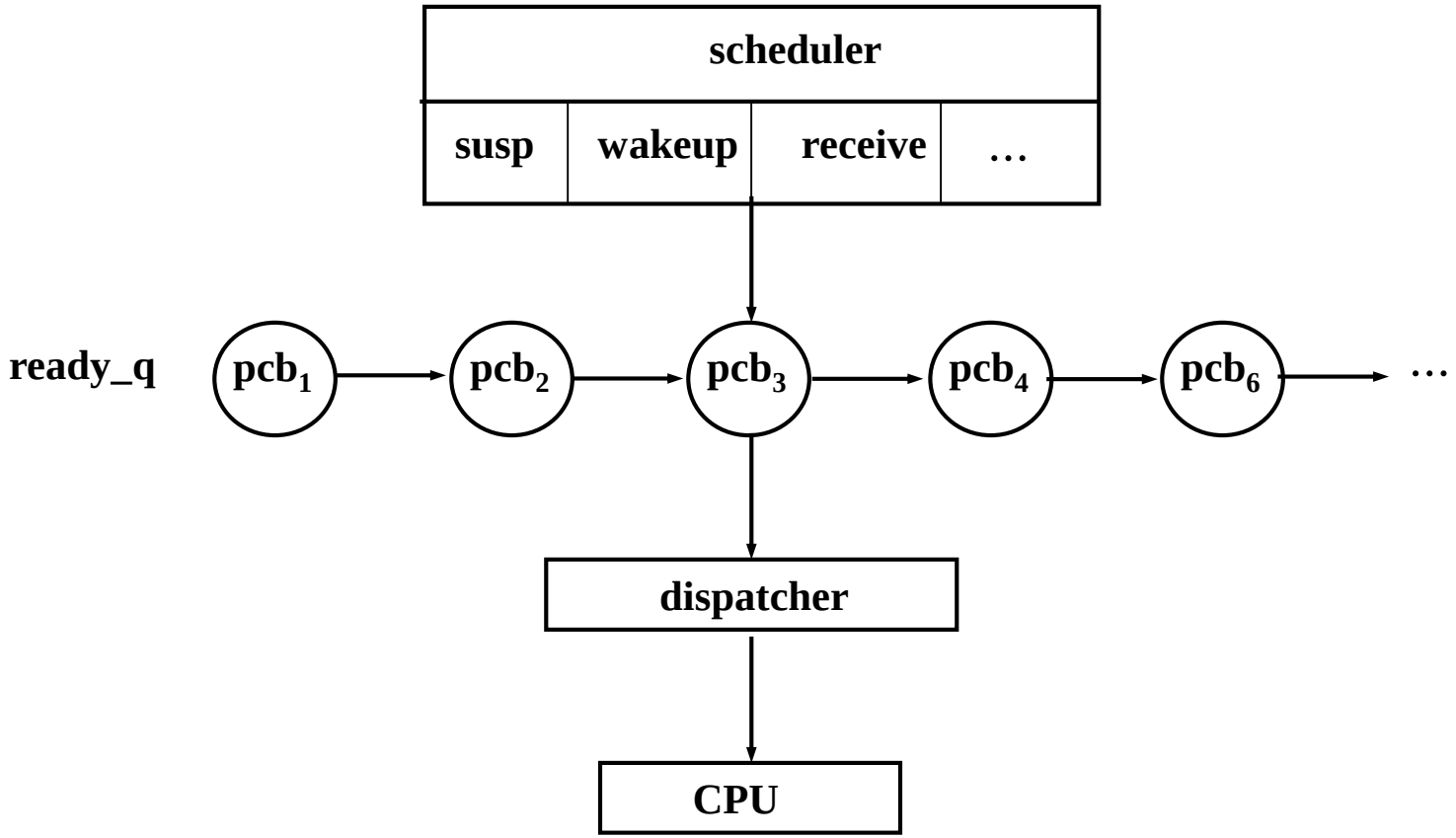
(1) 调度

在众多处于就绪状态的进程中，按一定的原则选择一个进程。

(2) 分派

当处理机空闲时，移出就绪队列中第一个进程，并赋予它使用处理机的权利。

(3) 调度分派结构图



调度/分派结构示意图

2. 进程调度的功能

(1) 进程管理的数据结构

(2) 决定调度策略

① 优先调度

就绪队列按进程优先级高低排序

② 先来先服务

就绪队列按进程来到的先后次序排序

(3) 实施处理机的分配和回收

3. 进程调度的方式

(1) 什么是调度方式

当一进程正在处理机上执行时，若有某个更为“重要而紧迫”的进程需要运行，系统如何分配处理机。

(2) 非剥夺方式

当“重要而紧迫”的进程来到时，让正在执行的进程继续执行，直到该进程完成或发生某事件而进入“完成”或“阻塞”状态时，才把处理机分配给“重要而紧迫”的进程。

(3) 剥夺方式

当“重要而紧迫”的进程来到时，便暂停正在执行的进程，立即把处理机分配给优先级更高的进程。

4. 进程调度算法

(1) 进程优先数调度算法

① 什么是进程优先数调度算法

预先确定各进程的优先数，系统把处理机的使用权赋予就绪队列中具备最高优先权 (优先数和一定的优先级相对应) 的就绪进程。

② 优先数的分类及确定

i **静态优先数**

在进程被创建时确定，且一经确定后在整个进程运行期间不再改变。

ii **静态优先数的确定**

- 优先数根据进程所需使用的资源来计算
- 优先数基于程序运行时间的估计
- 优先数基于进程的类型

iii **动态优先数**

进程优先数在进程运行期间可以改变。

iv **动态优先数的确定**

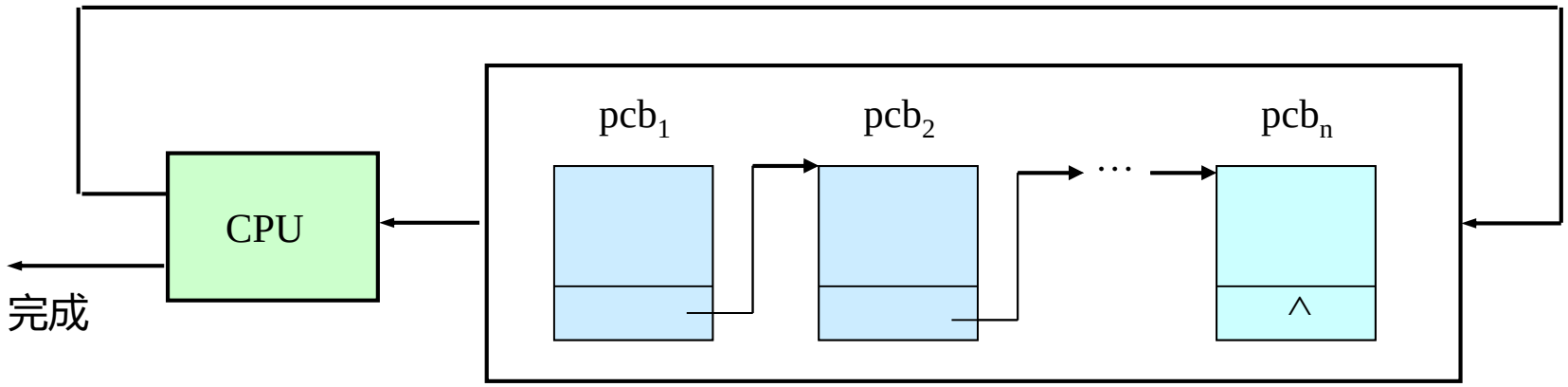
- 进程使用CPU超过一定数值时，降低优先数
- 进程I/O操作后，增加优先数
- 进程等待时间超过一定数值时，提高优先数定

(2) 循环轮转调度算法

① 什么是循环轮转调度算法

当CPU空闲时，选取就绪队列首元素，赋予一个时间片，当时间片用完时，该进程转为就绪态并进入就绪队列末端。

该队列排序的原则是什么？



简单循环轮转调度算法示意图

② 简单循环轮转调度算法

就绪队列中的所有进程以等速度向前进展。

$q = t/n$

t 为响应时间，n为进入系统的进程数目

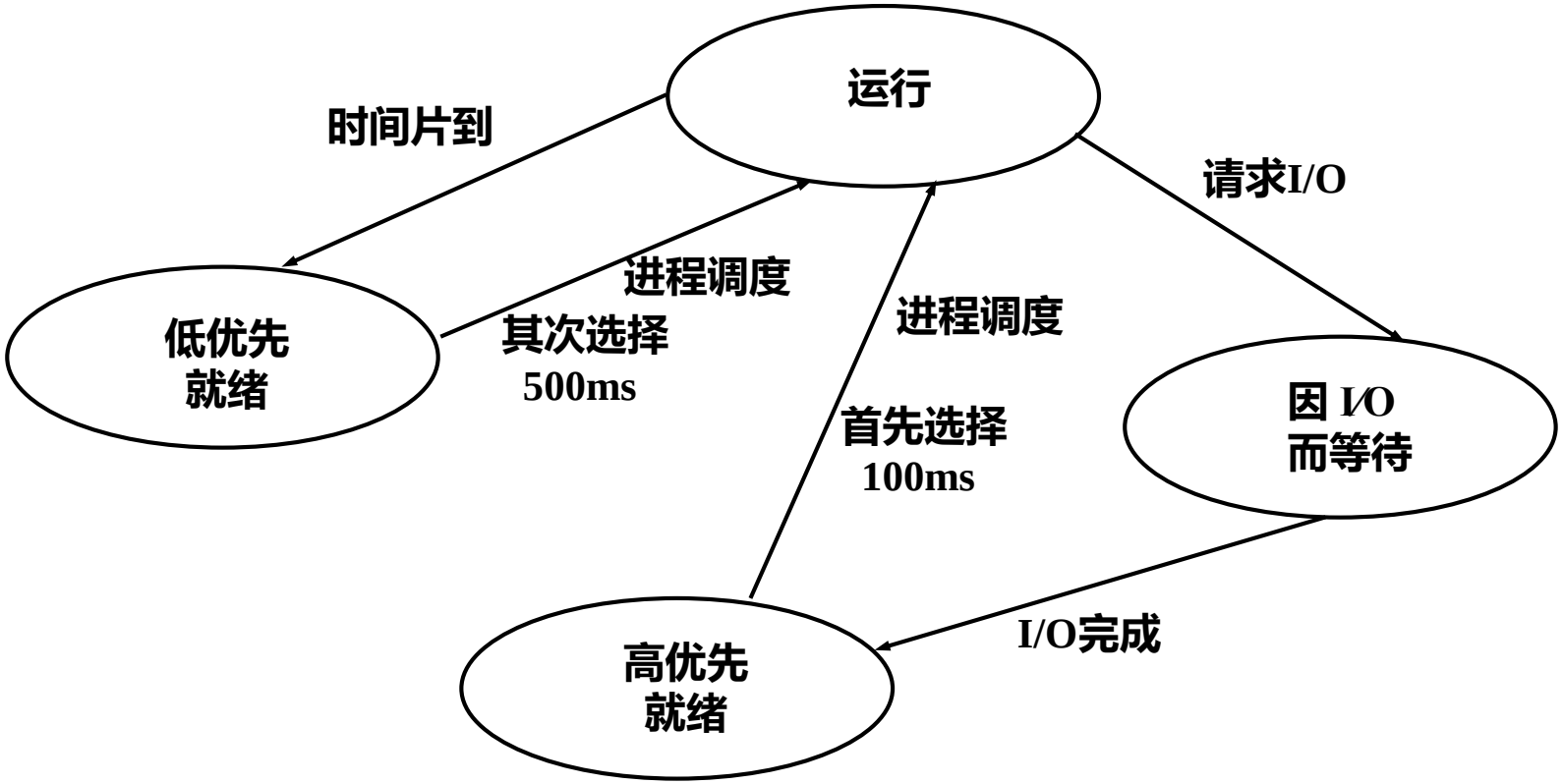
q 值的影响?

③ 循环轮转调度算法的发展

- 可变时间片轮转调度
- 多重时间片循环调度

5. 调度用的进程状态变迁图

(1) 一个调度用的进程状态变迁图的实例



调度用的进程状态变迁图

(2) 调度用进程状态变迁图实例的分析

① 进程状态

- 运行状态
- 低优先就绪状态
- 高优先就绪状态
- 因I/O而等待状态

② 队列结构

- 低优先就绪队列
- 高优先就绪队列
- 因I/O而等待队列

③ 进程调度算法

优先调度与时间片调度相结合的调度算法

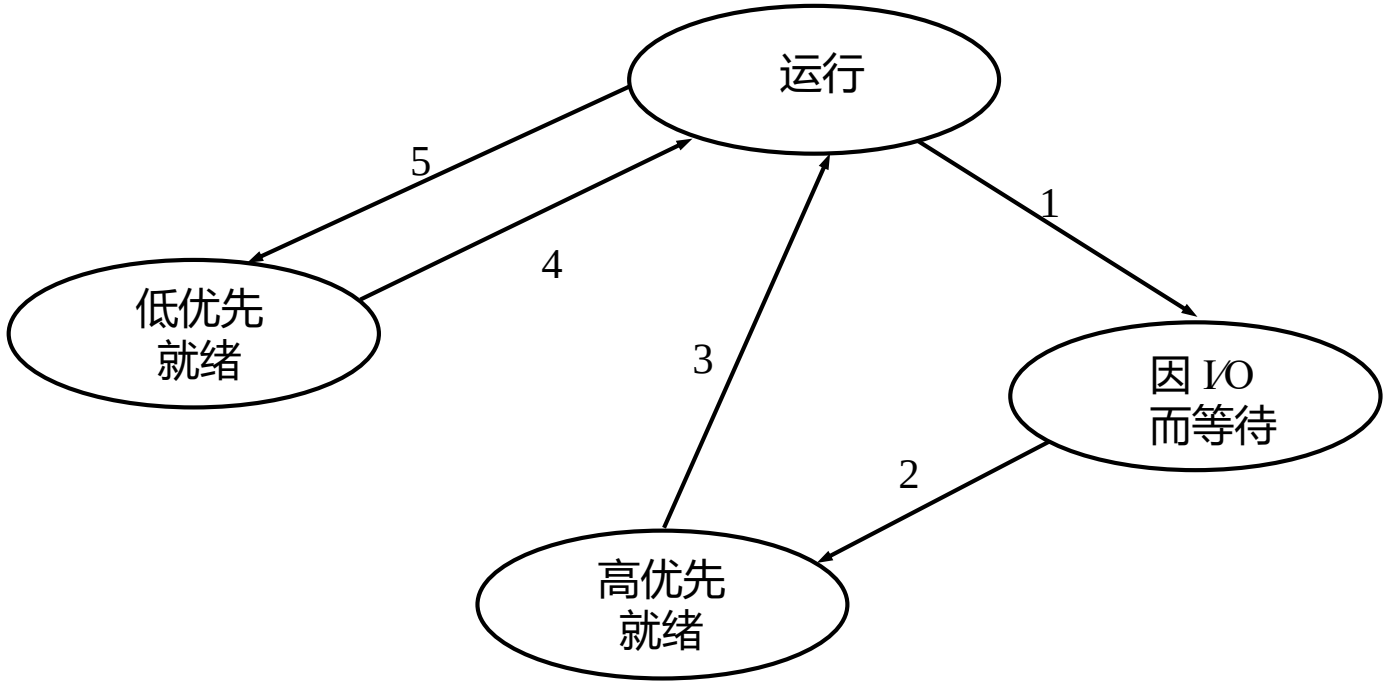
i 当CPU空闲时，若高优先就绪队列非空，则从高优先就绪队列中选择一个进程运行，分配时间片为100ms。

ii 当CPU空闲时，若高优先就绪队列为空，则从低优先就绪队列中选择一个进程运行，分配时间片为500ms。

④ 调度效果

优先照顾I/O量大的进程；适当照顾计算量大的进程。

(3) 较复杂进程状态的讨论



进程状态变迁图

变迁1 → 变迁3

变迁1 → 变迁4

变迁2 → 变迁3

进程及进程管理——小结

进程概念（掌握）

- 进程引入
  - 程序的顺序执行　定义　特点
  - 程序的并发执行　定义　特点
- 进程定义
  - 定义
  - 进程与程序的区别
- 进程状态
  - 三个基本状态、状态变迁图
  - 不同操作系统类型的进程状态变迁图
- 进程描述
  - PCB的定义与作用
  - 进程的组成
- 线程定义

进程及进程管理——小结

- **进程控制（理解）**
  - 进程控制原语
    - 基本进程控制原语
    - 进程控制原语的执行与进程状态的变化
  - 进程创建、进程撤销原语的功能
  - 进程等待、进程唤醒原语的功能
- **进程的相互制约关系（掌握）**
  - 进程互斥
    - 临界资源
    - 互斥
    - 临界区

进程及进程管理——小结

- 进程同步
  - 进程同步的概念
  - 进程同步的例

- **进程同步机构（掌握）**
  - 锁、上锁原语、 开锁原语
  - 信号灯及P、V操作

- **进程同步与互斥的实现（掌握）**
  - 用信号灯的P、V操作实现进程互斥
  - 两类同步问题的解答
    - 合作进程的执行次序
    - 共享缓冲区的合作进程的同步
  - 生产者——消费者问题及解答

进程及进程管理——小结

- **操作系统的并发控制机制（掌握）**
  - 创建进程、创建线程及其使用
  - 等待进程、线程的终止及其使用
  - 信号量与使用方法
  - 共享内存与使用方法

- **进程调度（掌握）**
  - 进程调度的功能
  - 调度方式　非剥夺方式　剥夺方式
  - 常用的进程调度算法
  - 调度用的进程状态变迁图的分析