

NEW PLAN

软考高级架构师

• — 段 新 征 程 •

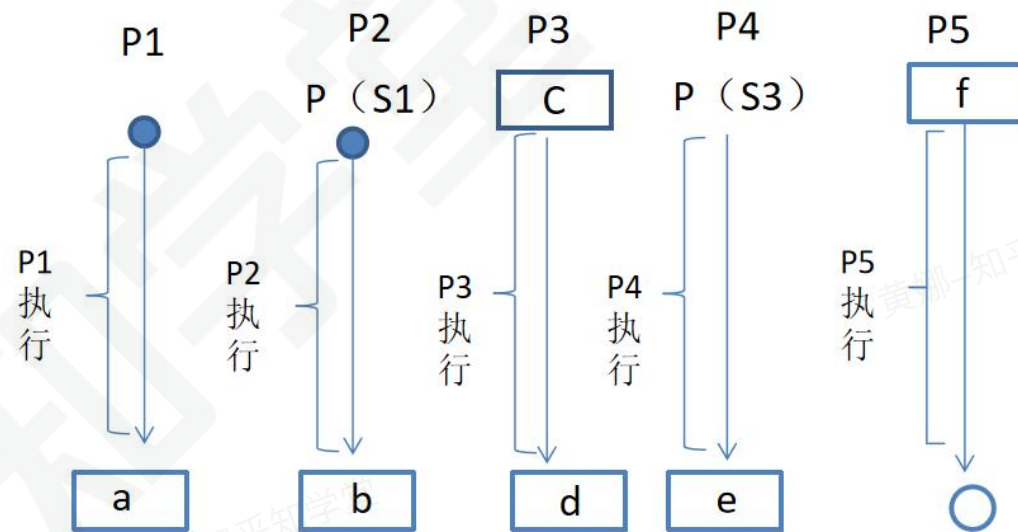
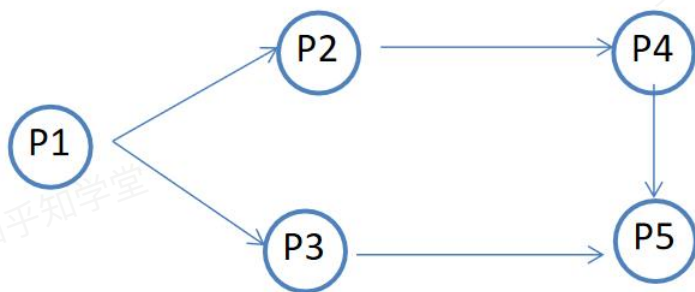


大纲介绍



进程管理

例：进程P1、P2、P3、P4和P5的前趋图如下图所示：



若用PV操作控制进程P1、P2、P3、P3、P4和P5并发执行的过程，则需要设置5个信号S1、S2、S3、S4和S5，且信号量S1~S5的初值都等于零。下图中a和b处应分别填(26)；c和d处应分别填写(27)；e和f处应分别填写(28)。

26、

- A. V(S1)P(S2)和V(S3)
- C. V(S1)V(S2)和V(S3)

- B. P(S1)V(S2)和V(S3)
- D. P(S1)P(S2)和V(S3)

27、

- A. P(S2)和P(S4)

- B. P(S2)和V(S4)

- C. V(S2)和P(S4)

- D. V(S2)和V(S4)

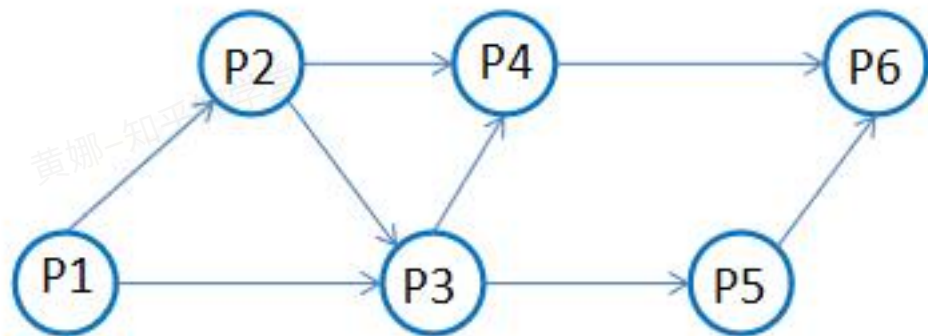
28、

- A. P(S4)和V(S4)V(S5)
- C. V(S3)和V(S4)V(S5)

- B. V(S5)和P(S4)P(S5)
- D. P(S3)和P(S4)V(P5)

进程管理

进程P1、P2、P3、P4、P5和P6的前趋图如下所示，若用PV操作控制这6个进程的同步与互斥的程序如下，那么程序中的空①和空②处应分别为(); 空③和空④处应分别为(); 空⑤和空⑥处应分别为()。



```
begin
S1,S2,S3,S4,S5,S6,S7,S8:semaphore;    //定义信号量
S1:=0; S2:=0; S3:=0; S4:=0; S5:=0; S6:=0; S7:=0; S8:=0;
Cobegin
Process P1   Process P2   Process P3   Process P4   Process P5   Process P6
begin        begin        begin        begin        begin        Begin
P1 执行;    ②;           P(S2);    P(S4);    P(S6);    ⑥;
①;         V(S3);        P3 执行;  P(S5);    P5 执行;    P6 执行;
end;        V(S4);        ④;         ⑤;         V(S8);
Coend;      end;        end;        end;        end;        end
end.
```

- A.V(S1)V(S2)和P(S2)
C.V(S1)V(S2)和P(S1)

- B.P(S1)P(S2)和V(S2)
D.P(S1)P(S2)和V(S1)

- A.V(S3)和V(S5)V(S6)
C.V(S3)和P(S5)P(S6)

- B.P(S3)和V(S5)V(S6)
D.P(S3)和P(S5)P(S6)

- A.P(S6)和P(S7)V(S8)
C.P(S6)和P(S7)P(S8)

- B.V(S6)和V(S7)V(S8)
D.V(S7)和P(S7)P(S8)

进程管理

假设某系统采用非抢占式优先级调度算法，若该系统有两个优先级相同的进程P1和P2，各进程的程序段如下所示，若信号量S1和S2的初值都为0。进程P1和P2并发执行后a、b和c的结果分别为：a=()，b=()，c=()。

P1 程序段

```
begin{  
    a:=1;  
    a:=a+2;  
    V (S1);  
    c:=a+1;  
    P (S2);  
    a:=a+c;  
}  
end
```

P2 程序段

```
begin{  
    b:=1;  
    b:=b+2;  
    P (S1);  
    b:=a+b;  
    V (S2);  
    c:=b+c;  
}  
end
```

(50) A.6 B.7 C.10 D.13

(51) A.4 B.6 C.9 D.10

(52) A.4 B.6 C.10 D.13



进程管理-死锁

当一个进程在等待永远不可能发生的事件时，就会产生死锁，若系统中有多进程处于死锁状态，就会造成系统死锁。

死锁产生的四个必要条件：

- 资源互斥
- 每个进程占有资源并等待其他资源
- 系统不能剥夺进程资源
- 进程资源图是一个环路。



进程管理-死锁

死锁产生后，解决措施是打破四大条件，有下列方法：

- **死锁预防**：采用某种策略限制并发进程对于资源的请求，破坏死锁产生的四个条件之一，使系统任何时刻都不满足死锁的条件。
- **死锁避免**：一般采用**银行家算法**来避免，银行家算法，就是提前计算出一条不会死锁的资源分配方法，才分配资源，否则不分配资源，相当于借贷，考虑对方还得起才借钱，提前考虑好以后，就可以避免死锁。
- **死锁检测**：允许死锁产生，但系统定时运行一个检测死锁的程序，若检测到系统中发生死锁，则设法加以解除。
- **死锁解除**：即死锁发生后的解除方法，如强制剥夺资源，撤销进程等。

死锁计算问题：系统内有 n 个进程，每个进程都需要 R 个资源，那么其发生死锁的最大资源数为 $n*(R-1)$ 。其不发生死锁的最小资源数为 $n*(R-1)+1$ 。



进程管理

例：某系统中有3个并发进程竞争资源R，每个进程都需要5个R，那么至少有()

A.12

B.13

C.14

D.15

进程管理

银行家算法真题：假设系统中有三类互斥资源R1、R2和R3，可用资源数分别为10、5和3。在T0时刻系统中有P1、P2、P3、P3和P5五个进程，这些进程对资源的最大需求和已分配资源数如下表所示，此时系统剩余的可用资源数分别为(27)。如果进程按(28)序列执行，那么系统状态是安全的。

A. 1、1和0 B. 1、1和1 C. 2、1和0 D. 2、0和1

A. P1—P2—P4—P5—P3 B. P5—P2—P4—P3—P1
C. P4—P2—P1—P5—P3 D. P5—P1—P4—P2—P3

资源 进程	最大需求量			已分配资源数		
	R1	R2	R3	R1	R2	R3
P1	5	3	1	1	1	1
P2	3	2	0	2	1	0
P3	6	1	1	3	1	0
P4	3	3	2	1	1	1
P5	2	1	1	1	1	0

进程管理-线程

传统的进程有两个属性：

- 可拥有资源的独立单位
- 可独立调度和分配的基本单位

引入线程后，**线程是独立调度的最小单位**，**进程是拥有资源的最小单位**，线程可以共享进程的公共数据、全局变量、代码、文件等资源，但不能共享线程独有的资源，如线程的栈指针等标识数据。

THE END

功不唐捐，玉汝于成！

• 开 启 新 征 程 •

