



南京大学  
NANJING UNIVERSITY

# 计算机与操作系统

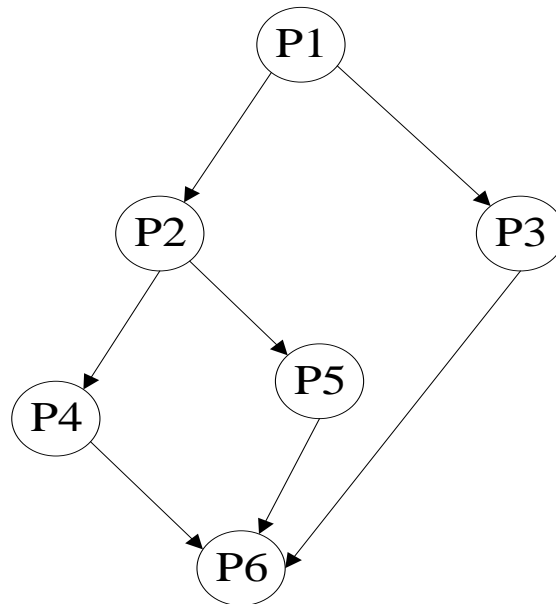
## 第十一讲 并发程序设计

### 习题讲解

南京大学软件学院



# 信号量 - 前驱关系



类似于PERT图



南京大學

NANJING UNIVERSITY

# 信号量 - 前驱关系

```
* Semaphore s1=0; /*表示进程P1是否已经执行完成*/
* Semaphore s2=0; /*表示进程P2是否已经执行完成*/
* Semaphore s3=0; /*表示进程P3是否已经执行完成*/
* Semaphore s4=0; /*表示进程P4是否已经执行完成*/
* Semaphore s5=0; /*表示进程P5是否已经执行完成*/
* main () {
* cobegin
*   * P1();
*   * P2();
*   * P3();
*   * P4();
*   * P5();
*   * p6();
* coend
* }
```



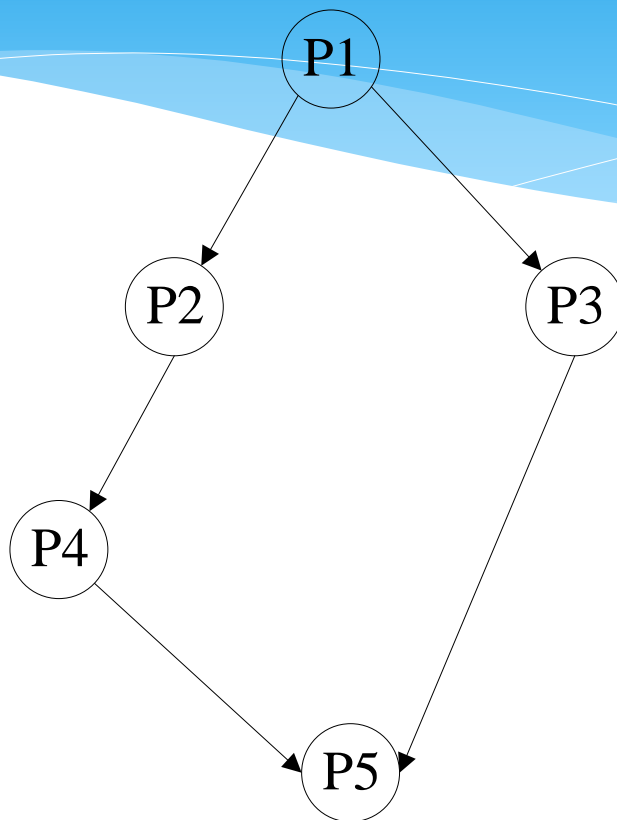
# 信号量 - 前驱关系

<b>P1( )</b> { .... <b>V(s1)</b> <b>V(s1)</b> }	<b>P3( )</b> { <b>P(s1)</b> .... <b>V(s3)</b> }	<b>P5( )</b> { <b>P(s2)</b> .... <b>V(s5)</b> }
<b>P2( )</b> { <b>P(s1)</b> .... <b>V(s2)</b> <b>V(s2)</b> }	<b>P4( )</b> { <b>P(s2)</b> .... <b>V(s4)</b> }	<b>P6( )</b> { <b>P(s3)</b> <b>P(s4)</b> <b>P(s5)</b> .... }



南京大学  
NANJING UNIVERSITY

# 信号量 - 前驱关系



类似于PERT图



南京大学  
NANJING UNIVERSITY

# 习题

## (信号量与PV操作)

- \* 1、读者写者问题
- \* 2、睡眠的理发师问题
- \* 3、农夫猎人问题
- \* 4、银行业务问题
- \* 5、缓冲区管理
- \* 6、售票问题
- \* 7、吸烟者问题



# 1、读者/写者问题

- \* 读者与写者问题(reader-writer problem) (Courtois, 1971)也是一个经典的并发程序设计问题。有两组并发进程：读者和写者，共享一个文件F，要求：
- \* (1)允许多个读者可同时对文件执行读操作
- \* (2)只允许一个写者往文件中写信息
- \* (3)任一写者在完成写操作之前不允许其他读者或写者工作
- \* (4)写者执行写操作前，应让已有的写者和读者全部退出
- \* 使用PV操作求解该问题

# 读者/写者问题

```
semaphore rmutex, wmutex;  
    rmutex=1; wmutex=1; S=1; //增加互斥信号量S  
int readcount=0; //读进程计数
```

```
process reader_i( ) {  
while (true) {  
    {  
        P(rmutex);  
        if (readcount==0) P(wmutex);  
        readcount++;  
    }  
    V(rmutex);  
    读文件;  
    {  
        P(rmutex);  
        readcount--;  
        if(readcount==0) V(wmutex);  
    }  
    V(rmutex);  
}  
}
```

```
process writer_i( ) {  
while(true) {  
    {  
        P(wmutex);  
        写文件;  
    }  
    V(wmutex);  
}  
}
```

? 什么问题  
读者优先!



信号量解决读者写者问题：写者优先

```
semaphore rmutex, wmutex, S;  
    rmutex=1; wmutex=1; S=1; //增加互斥信号量S  
int readcount=0; //读进程计数
```

```
process reader_i( ) {  
    while (true) {  
        P(S);  
        P(rmutex);  
        if (readcount==0) P(wmutex);  
        readcount++;  
        V(rmutex);  
        V(S);  
        读文件;  
        P(rmutex);  
        readcount--;  
        if(readcount==0) V(wmutex);  
        V(rmutex);  
    }  
}
```

```
process writer_i( ) {  
    while(true) {  
        P(S);  
        P(wmutex);  
        写文件;  
        V(wmutex);  
        V(S);  
    }  
}
```

# 读者/写者问题 (写者优先)

```
int readcount = 0, writecount = 0;  
semaphore x=1, y=1, z=1;    // readcount, writecount 互斥  
semaphore rmutex=1, wmutex=1; // 读锁, 写锁
```

process reader

```
{  
  P(z);  
  {  
    P(rmutex);  
    {  
      P(x);  
      readcount++;  
      if (readcount==1) P(wmutex);  
      V(x);  
    }  
    V(rmutex);  
  }  
  V(z);  
  read;  
  {  
    P(x);  
    readcount--;  
    if (readcount==0) V(wmutex);  
  }  
  V(x);  
};
```

process writer

```
{  
  P(y);  
  writecount++;  
  if (writecount==1) P(rmutex);  
  V(y);  
  {  
    P(wmutex);  
    write;  
    V(wmutex);  
  }  
  P(y);  
  writecount--;  
  if (writecount==0) V(rmutex);  
  V(y);  
};
```



南京大學

NANJING UNIVERSITY

## 2、睡眠的理发师问题

- \* 理发店理有一位理发师、一把理发椅和 $n$ 把供等候理发的顾客坐的椅子
- \* 如果没有顾客，理发师便在理发椅上睡觉
- \* 一个顾客到来时，它必须叫醒理发师
- \* 如果理发师正在理发时又有顾客来到，则如果有空椅子可坐，就坐下来等待，否则就离开
- \* 使用PV操作求解该问题



# 睡眠的理发师问题

- \* `int waiting=0;`//等候理发顾客坐的椅子数
- \* `int CHAIRS=N;` //为顾客准备的椅子数
- \* `semaphore customers,barbers,mutex;`
- \* `customers=0;barbers=0;mutex=1;`



# 睡眠的理发师问题

```
int waiting=0;           //等候理发顾客坐的椅子数
int CHAIRS=N;            //为顾客准备的椅子数
semaphore customers, barbers, mutex;
customers=0; barbers=0; mutex=1;
```

```
process barber() {
while(true) {
    P(customers);
    //有顾客吗?若无顾客,理发师睡眠
    P(mutex);
    //若有顾客时,进入临界区
    waiting--; //等候顾客数少一个
    V(barbers); //理发师准备为顾客理发
    V(mutex);   //退出临界区
    cut_hair();
    //理发师正在理发(非临界区)
    }
}
```

```
process customer_i() {
    P(mutex);           //进入临界区
    if(waiting<CHAIRS) {
        //有空椅子吗
        waiting++; //等候顾客数加1
        V(customers); //唤醒理发师
        V(mutex);   //退出临界区
        P(barbers);
        //理发师忙,顾客坐下等待
        get_haircut(); //否则顾客坐下理发
    }
    else V(mutex); //人满了,走吧!
}
```



南京大學

NANJING UNIVERSITY

### 3、农夫猎人问题

- \* 有一个铁笼子，每次只能放入一个动物。猎手向笼中放入老虎，农夫向笼中放入羊；动物园等待取笼中的老虎，饭店等待取笼中的羊。请用P、V操作原语写出同步执行的程序



# 农夫猎人问题

```
semaphore Scage=1;  
semaphore Stiger=0;  
semaphore Ssheep=0;
```

```
void hunter()  
{  
    while (true) {  
        .....  
        P(Scage);  
        将虎放入笼中;  
        V(Stiger);  
    }  
}
```

```
void peasant()  
{  
    while (true) {  
        .....  
        P(Scage);  
        将羊放入笼中;  
        V(Ssheep);  
    }  
}
```

```
void hotel()  
{  
    while (true) {  
        P(Ssheep);  
        将羊取出笼中;  
        V(Scage);  
        .....  
    }  
}
```

```
void zoo()  
{  
    while (true){  
        P(Stiger);  
        将虎取出笼中;  
        V(Scage);  
        .....  
    }  
}
```

```
void main()  
{  
    parbegin(hunter, peasant, hotel, zoo);  
}
```



南京大學

NANJING UNIVERSITY

## 4、银行业务问题

- \* 某大型银行办理人民币储蓄业务，由 $n$ 个储蓄员负责。每个顾客进入银行后先至取号机取一个号，并且在等待区找到空沙发坐下等着叫号。取号机给出的号码依次递增，并假定有足够多的空沙发容纳顾客。当一个储蓄员空闲下来，就叫下一个号。请用信号量和 $P$ ， $V$ 操作正确编写储蓄员进程和顾客进程的程序





# 银行业务问题

```
var customer_count, server_count, mutex: semaphore;  
customer_count:=0; server_count:=n;  
mutex:=1;
```

```
process customeri(i=1,2,....)  
begin  
  take a number;  
  P(mutex);  
  等待区找到空沙发坐下;  
  V(mutex);  
  V(customer_count);  
  P(server_count);  
end;
```

```
Process servers j(j=1,2,3,...)  
Begin  
  L: P(customer_count);  
  P(mutex);  
  被呼号顾客离开沙发走出等待区;  
  V(mutex);  
  为该号客人服务;  
  客人离开;  
  V(server_count);  
  go to L;  
end;
```



## 5、缓冲区管理

- \* 有 $n$ 个进程将字符逐个读入到一个容量为80的缓冲区中( $n > 1$ )，当缓冲区满后，由输出进程 $Q$ 负责一次性取走这80个字符。这种过程循环往复，请用信号量和 $P$ 、 $V$ 操作写出 $n$ 个读入进程( $P_1, P_2, \dots, P_n$ )和输出进程 $Q$ 能正确工作的动作序列



# 缓冲区管理

```
var mutex,empty,full:semaphore;  
count,in:integer  
buffer:array[0..79] of char;  
mutex=1;empty=80;full=0;  
count=0;in=0;
```

```
process Pi(i=1,...,n))  
begin  
  L: 读入一字符到x;  
  P(empty);  
  P(mutex);  
  Buffer[in]=x;  
  in=(in+1) % 80;  
  count++;  
  if (count==80)  
    {count=0; V(mutex); V(full); }  
  else V(mutex);  
  goto L;  
end;
```

```
process Q  
begin  
  while(true) {  
    P(full);  
    P(mutex);  
    for(int j=0; j< 80;j++)  
      read buffer[j];  
    in:=0;  
    V(mutex);  
    for (int j=0; j< 80;j++)  
      V(empty);  
  }  
end;
```



## 6、售票问题

- \* 汽车司机与售票员之间必须协同工作，一方面只有售票员把车门关好了司机才能开车，因此，售票员关好门应通知司机开车，然后售票员进行售票。另一方面，只有当汽车已经停下，售票员才能开门上客，故司机停车后应该通知售票员。假定某辆公共汽车上有一名司机与两名售票员，汽车当前正在始发站停车上客，试用信号量与P、V操作写出他们的同步算法



# 售票问题

```
Var run1, run2, stop1, stop2: semaphore;  
run1:=0; run2:=0; stop1:=0; stop2:=0;
```

```
void Driver() {  
    while (true)  
    {  
        P(run1);  
        P(run2);  
        开车;  
        停车;  
        V(stop1);  
        V(stop2);  
    }  
}
```

```
void Seller1() {  
    while (true) {  
        上乘客;  
        关车门;  
        V(run1);  
        售车票;  
        P(stop1);  
        开车门;  
        下乘客;  
    }  
}
```

```
void Seller2() {  
    while (true) {  
        上乘客;  
        关车门;  
        V(run2);  
        售车票;  
        P(stop2);  
        开车门;  
        下乘客;  
    }  
}
```

```
void main() {  
    parbegin(Driver; Seller1; Seller2);  
}
```



## 7、吸烟者问题

- \* 一个经典同步问题：吸烟者问题(patil, 1971)。三个吸烟者在一个房间内，还有一个香烟供应者。为了制造并抽掉香烟，每个吸烟者需要三样东西：烟草、纸和火柴，供应者有丰富货物提供。三个吸烟者中，第一个有自己的烟草，第二个有自己的纸和第三个有自己的火柴。供应者随机地将两样东西放在桌子上，允许一个吸烟者进行对健康不利的吸烟。当吸烟者完成吸烟后唤醒供应者，供应者再把两样东西放在桌子上，唤醒另一个吸烟者。试用信号量和P、V操作求解该问题



# 吸烟者问题

```
semaphor:s0,s1,s2,s3;  
S0=1;S1=0;S2=0;S3=0;
```

Process businessman {

//供应者进程

L1: i:=RAND() mod 3;

j:=RAND() mod 3;

If (i=j) then goto L1;

P(S0):

Put\_items [i]\_on\_table;

Put\_items [j]\_on\_table;

if (i=0 and j=1) or (i=1 and j=0) V(S[3]):

if (i=1 and j=2) or (i=2 and j=1) V(S[1]):

if (i=0 and j=2) or (i=2 and j=0)

V(S[2]);

goto L1;

}

Process consumer (k) {

//吸烟者进程, k=1,2,3

L1:

P(S[k]);

take\_one\_item\_from\_table;

take\_one\_item\_from\_table;

V(S0):

make\_cigarette\_and\_smokeing

goto L1;

}