操作系统　Operating System

第四章 进程与并发程序设计(4)
——经典的进程同步与互斥问题

沃天宇
woty@buaa.edu.cn
2024年4月17日

The Institute of Advanced Computing Technology

内容提要

- 同步与互斥问题
- 基于忙等待的互斥方法
- 基于信号量的方法
- 基于管程的同步与互斥
- 进程通信的主要方法
- 经典的进程同步与互斥问题

The Institute of Advanced Computing Technology

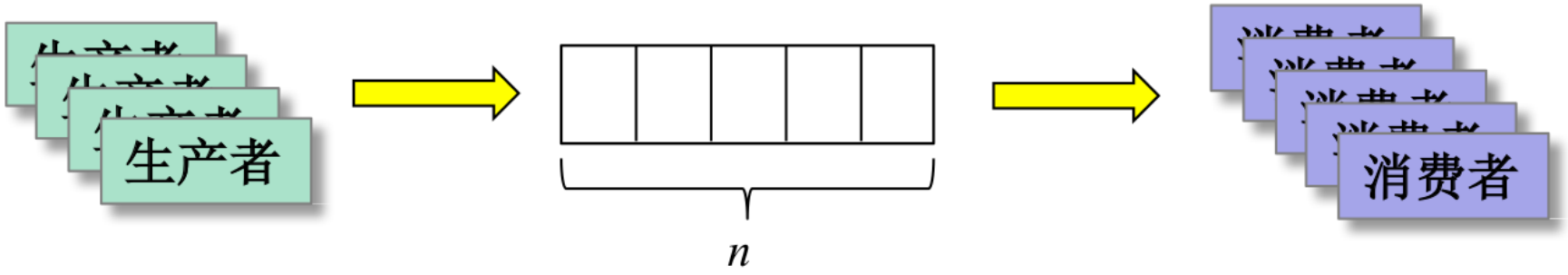
经典进程同步问题

- 生产者－消费者问题(the producer-consumer problem)
- 读者－写者问题(the readers-writers problem)
- 哲学家进餐问题(the dining philosophers problem)

The Institute of Advanced Computing Technology

生产者—消费者问题

- 问题描述：若干进程通过有限的共享缓冲区交换数据。其中，“生产者”进程不断写入，而“消费者”进程不断读出；共享缓冲区共有N个；任何时刻只能有一个进程可对共享缓冲区进行操作。



The Institute of Advanced Computing Technology

生产者—消费者问题

- 两个隐含条件：
  1. 消费者和生产者数量不固定。
  2. 消费者和生产者不能同时使用缓存区。
- 行为分析：
  - 生产者：生产产品，放置产品(有空缓冲区)。
  - 消费者：取出产品(有产品)，消费产品。
- 行为关系：
  - 生产者之间:互斥(放置产品)
  - 消费者之间:互斥(取出产品)
  - 生产者与消费者之间：互斥(放/取产品) 同步(放置——取出)

The Institute of Advanced Computing Technology

用PV操作解决生产者/消费者问题

- 信号量设置：

  semaphore mutex =1;　　//互斥

  semaphore empty=N;　　//空闲数量

  semaphore full=0;　　//产品数量

- 实际上，full和empty是同一个含义：

  full + empty == N

The Institute of Advanced Computing Technology

用PV操作解决生产者/消费者问题

生产者

```
P(empty);
P(mutex);
    one >> buffer
V(mutex)
V(full)
```

消费者

```
P(full);
P(mutex);
    one << buffer
V(mutex)
V(empty)
```

The Institute of Advanced Computing Technology

用PV操作解决生产者/消费者问题

生产者

```
P(mutex);
P(empty);
    one >> buffer
V(full)
V(mutex)
```

消费者

```
P(mutex);
P(full);
    one << buffer
V(empty)
V(mutex)
```

如果两个V交换顺序呢？

The Institute of Advanced Computing Technology

完整的伪代码写法

```
Semaphore full = 0;
Semaphore empty = n;
Semaphore mutex = 1;
ItemType buffer[0..n-1];
int in = 0, out =0;

main() {
    Cobegin
        producer();
        consumer();
    Coend
}
```

```
producer() {
    while(true){
        生产产品nextp;
        P(empty);
        P(mutex);
        buffer[in] = nextp;
        in = (in + 1) MOD n;
        V(mutex);
        V(full);
    }
}
```

```
consumer() {
    while(true){
        P(full);
        P(mutex);
        nextc = buffer[out];
        out = (out + 1) MOD n;
        V(mutex);
        V(empty);
        消费nextc中的产品
    }
}
```

用管程解决生产者消费者问题

```
monitor ProducerConsumer
    condition full, empty;
    integer count;

    procedure insert (item: integer);
    begin
        if count == N then wait(full);
        insert_item(item); count++;
        if count ==1 then signal(empty);
    end;

    function remove: integer;
    begin
        if count ==0 then wait(empty);
        remove = remove_item; count--;
        if count==N-1 then signal(full);
    end;

    count:=0;
end monitor;
```

```
procedure producer;
begin
    while true do
    begin

        item = produce_item;
        ProducerConsumer.insert(item);
    end
end;

procedure consumer;
begin
    while true do
    begin
        item=ProducerConsumer.remove;
        consume_item(item);

    end
end;
```

The Institute of Advanced Computing Technology

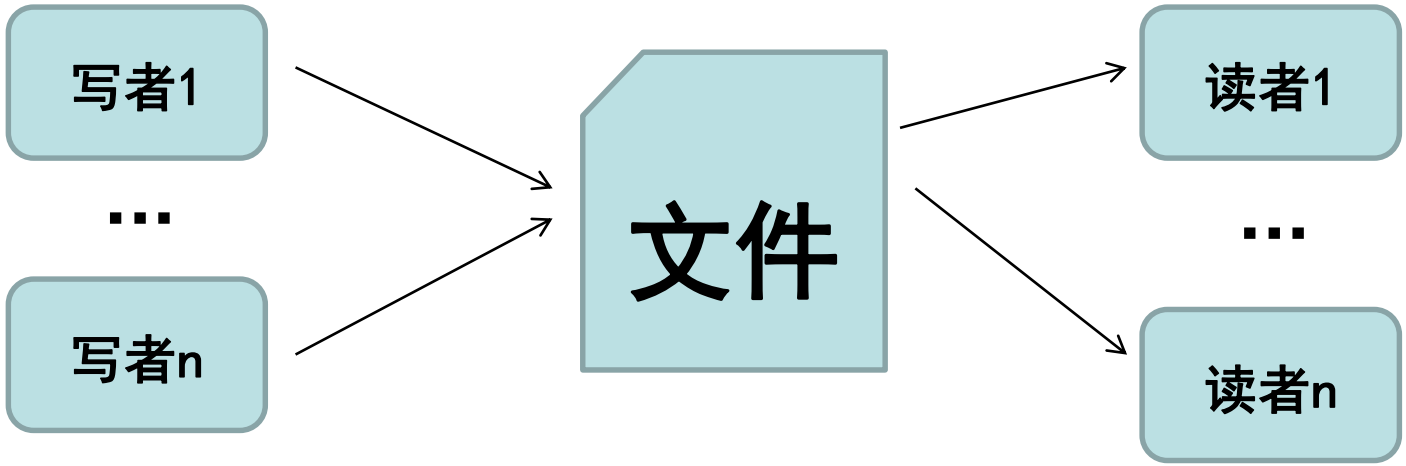
采用AND信号量集

- SP(empty, mutex),
- SP(full, mutex)
- 同学们可自行练习

The Institute of Advanced Computing Technology

读者－写者问题

- 问题描述：对共享资源的读写操作，任一时刻“写者”最多只允许一个，而“读者”则允许多个，即“读－写”互斥，“写－写”互斥，“读－读”允许



The Institute of Advanced Computing Technology

读者-写者问题分析

- 生活中的实例：12306定票
  - 读者：？
  - 写者：？
- 多个线程/进程共享内存中的对象
  - 有些进程读，有些进程写
  - 同一时刻，只有一个激活的写进程
  - 同一时刻，可以有多个激活的读进程
- 当读进程激活时，是否允许写进程进入？
- 当写进程激活时，？

The Institute of Advanced Computing Technology

采用信号量机制

- 基本框架

Writer

P(wmutex);
写数据
V(wmutex);

Reader

if first_reader? then P(wmutex);
读数据
if last_reader? then V(wmutex);

The Institute of Advanced Computing Technology

采用信号量机制

- wmutex表示“允许写”，初值是1。
- 公共变量readcount表示“正在读”的进程数，初值是0；
- mutex表示对readcount的互斥操作，初值是1。

The Institute of Advanced Computing Technology

Writer

```
P(wmutex);
    write
V(wmutex);
```

Reader

```
P(mutex);
if readcount=0 then  P(wmutex);
readcount := readcount +1;
V(mutex);
read
P(mutex);
readcount := readcount -1;
if readcount=0 then V(wmutex);
V(mutex);
```

- 如果有一个写者，第一个读者将阻塞，其他读者也会阻塞
- 如果一个读者激活，所有的读者都会陆续激活，哪个最先激活？
- 最后一个读者退出时，唤醒一个写者；如果没有写者呢？
- 如果读者和写者均阻塞在wmutex上，写者退出后，哪一个先激活？

完整伪代码，参见教材P128页。

The Institute of Advanced Computing Technology

Writer

```
P(wmutex);
    write
V(wmutex);
```

Reader

```
P(mutex);
if readcount=0 then  P(wmutex);
readcount := readcount +1;
V(mutex);
read
P(mutex)
readcount := readcount -1;
if readcount=0 then V(wmutex);
V(mutex)
```

回顾“读者-写者”问题

该算法是对读者有利，还是对写者有利？

The Institute of Advanced Computing Technology

之前（对读者有利）算法

- 从读者、写者自身角度总结几条规则
- 写者：
  - 开始后，没有其他写者、读者可以进入
  - 必须确保没有正在读的读者，才能开始写
  - 写完之后允许其他读写者进入
- 读者开始读之前需要确保：
  - 第一个读者开始读，到最后一个读者结束读之间不允许有写进程进入写过程

The Institute of Advanced Computing Technology

对写者有利的算法？

- 在之前的算法基础上增加什么？
  - 写者出现后，新读者不允许抢在该写者之前进行读操作。
  - 也就是说：写者出现后，后续读者不允许执行rcount=rcount+1
  - 因此需要增加一个互斥信号量

The Institute of Advanced Computing Technology

读写公平的算法

Writer

```
P(rwmutex);
P(wmutex);
写数据
V(wmutex);
V(rwmutex);
```

Reader

```
P(rwmutex);
P(mutex);
if readcount=0 then  P(wmutex);
readcount := readcount +1;
V(mutex);
V(rwmutex);
read
P(mutex);
readcount := readcount -1;
if readcount=0 then V(wmutex);
V(mutex);
```

注意这个位置
能不能放在read之后？

The Institute of Advanced Computing Technology

哲学家进餐问题 (the dining philosophers problem)

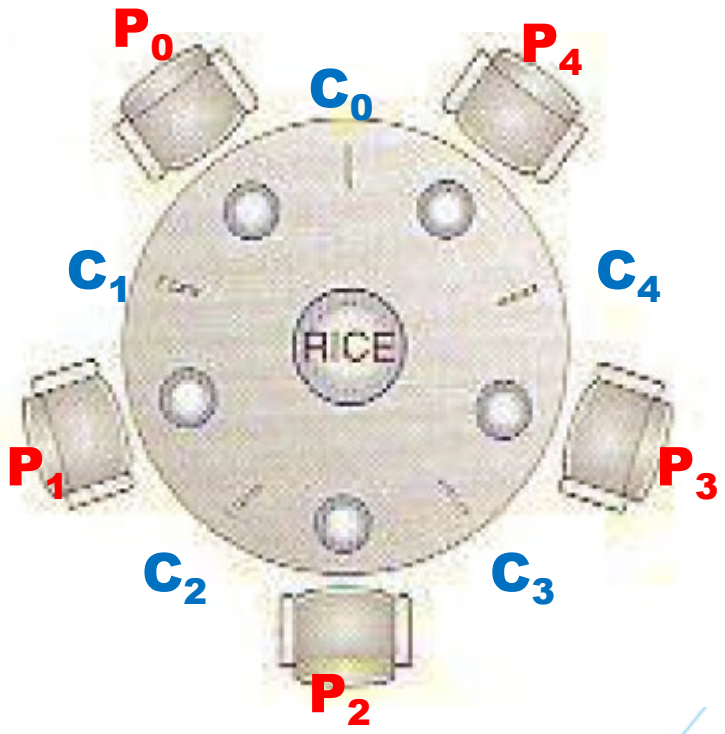
- 问题描述：（由Dijkstra首先提出并解决）5个哲学家围绕一张圆桌而坐，桌子上放着5支筷子，每两个哲学家之间放一支；
- 哲学家的动作包括思考和进餐，进餐时需要同时拿起他左边和右边的两支筷子，思考时则同时将两支筷子放回原处。
- 如何保证哲学家们的动作有序进行？如：不出现相邻者同时进餐；不出现有人永远拿不到筷子。

The Institute of Advanced Computing Technology

基于P-V操作的方法

```
Var chopstick : array[0..4] of semaphore;

P(chopstick[i]);
P(chopstick[(i+1)mod 5]);
    eat
V(chopstick[i]);
V(chopstick [(i+1)mod 5]);
   think
```



有没有什么问题？

The Institute of Advanced Computing Technology

哲学家就餐问题的解题思路

- 至多只允许四个哲学家同时（尝试）进餐，以保证至少有一个哲学家能够进餐，最终总会释放出他所使用过的两支筷子，从而可使更多的哲学家进餐。设置信号量room=4。（破除**资源互斥**）
- 对筷子进行编号，奇数号先拿左，再拿右；偶数号相反。（破除**循环等待**）
- 同时拿起两根筷子，否则不拿起。（破除**保持等待**）

The Institute of Advanced Computing Technology

基于AND信号量集的方法

```
Var chopstick : array[0..4] of semaphore;
    think
SP(chopstick[(i+1)mod 5], chopstick[i]);
    eat
SV(chopstick[(i+1)mod 5], chopstick[i]);
```

The Institute of Advanc

“生产者-消费者”扩展问题

- 设有一个可以装A、B两种物品的仓库,其容量无限大,但要求仓库中A、B两种物品的数量满足下述不等式:
  - -M≤A物品数量-B物品数量≤N
    - 即： -N≤B物品数量- A物品数量≤M
    - 即： A不能比B多N以上 AND B不能比A多M以上
  - 其中M和N为正整数.
- 试用信号量和PV操作描述A、B两种物品的入库过程.

The Institute of Advanced Computing Technology

算法描述

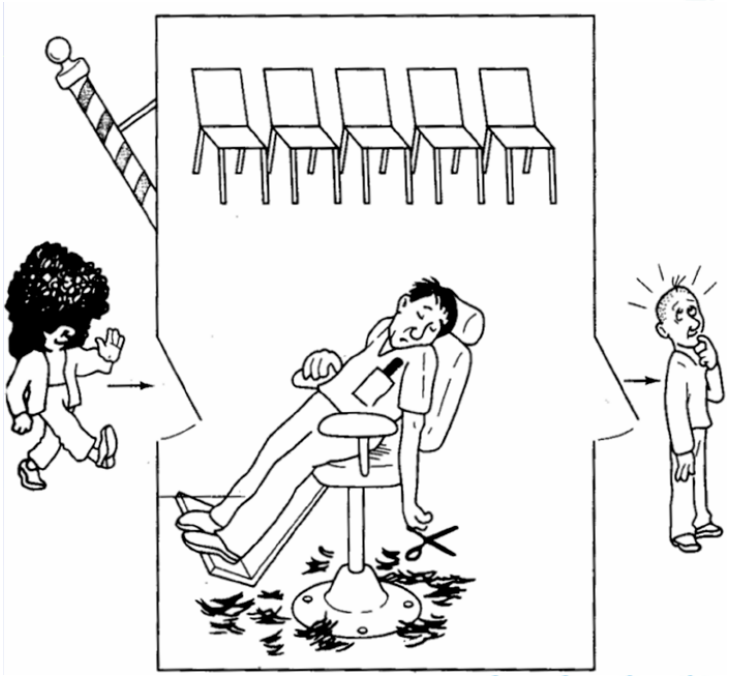
sa: A货物比B货物多的额度
sb: B货物比A货物多的额度

```
Semaphore mutex=1，sa=N，sb=M;
cobegin
    procedure A:                    procedure B:
      while(TURE)                       while(TURE)
        begin                            begin
          p(sa);                          p(sb);
          p(mutex);                      p(mutex);
          A产品入库；                          B产品入库；
          V(mutex);                      V(mutex);
          V(sb);                          V(sa);
        end                               end
coend
```

The Institute of Advanced Computing Technology

理发师问题

- 理发店里有1位理发师、1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**，当一个顾客到来时，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**。



互斥资源：理发师、顾客、椅子

同步约束：访问椅子、顾客唤醒理发师、理发师唤醒下一个位等待顾客

理发师问题

```
int waiting = 0;
```

Barber:

```
while(true) {
    睡觉

    开始理发
    理发
}
```

Custom_i:

```
if (waiting < N) {
    坐下来

    唤醒理发师
    等待理发
    接受理发
}
```

理发师问题

```
int waiting = 0;
semaphore mutex=1;
semaphore custormers=0;
semaphore barber=0;
```

Barber:

```
while(true) {
    P(custormers);
    P(mutex);
    waiting --;
    V(mutex);
    V(barber);
    CutHair();
}
```

Custom_i:

```
P(mutex);
if (waiting < N) {
    waiting ++;
    V(mutex);
    V(custormers);
    P(barber);
    GetCutHair();
}
else {
    V(mutex);
}
```

理发师问题变种

1. 【多理发师】理发店里有m位理发师，n把供等候理发的顾客坐的椅子；如果没有顾客，理发师便睡觉，当顾客到来时，叫醒理1位发师；如果所有理发师正在理发时，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。
2. 【店满等待】理发店里有m位理发师，n把供等候理发的顾客坐的椅子；如果没有顾客，理发师便睡觉，当顾客到来时，叫醒理1位发师；如果所有理发师正在理发时，又有顾客来到，则如果有空椅子可坐，就坐下来等待，否则就在门口等待。

The Institute of Advanced Computing Technology

吸烟者问题

- 三个吸烟者在一间房间内，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。供应者有丰富的货物提供。
- 三个吸烟者中，第一个有自己的烟草，第二个有自己的纸，第三个有自己的火柴。
- 供应者将两样东西放在桌子上，允许一个吸烟者吸烟。当吸烟者完成吸烟后唤醒供应者，供应者再放两样东西（随机地）在桌面上，然后唤醒另一个吸烟者。
- 试为吸烟者和供应者编写程序解决问题。

The Institute of Advanced Computing Technology

解题思路

- P：供应者；S1,S2,S3:吸烟者;
  a:烟；b:纸; c:火柴。
- 互斥资源：桌子t
- 同步进程：P与S1，P与S2，P与S3

```
P
P(t);
放东西;
if(b &c ) V(S1);
else if(a & c) V(S2);
else V(S3);
```

```
S1
P(S1);
取纸和火柴
吸烟
V(t);
```

…

The Institute of Advanced Computing Technology

小结

- 同步与互斥
- 基于忙等待的方法
- 基于信号量的方法
- 管程方法
- 经典同步与互斥问题
  - 生产者-消费者
  - 读者-写者
  - 哲学家就餐

The Institute of Advanced Computing Technology