操作系统A

Principles of Operating System

北京大学计算机科学技术系 陈向群

Department of computer science and Technology, Peking University

2020 Autumn

经典IPC问题讨论

课堂讨论安排

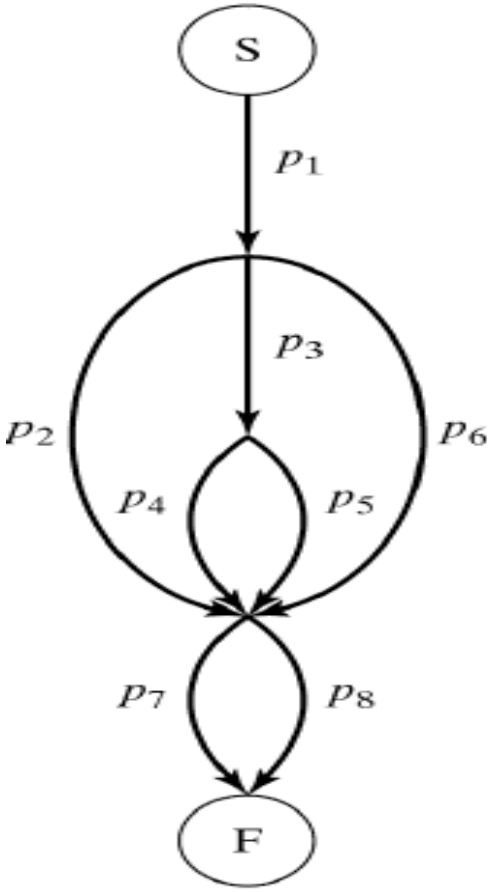
- 纯同步问题——
- 另类PV操作问题——
- 阅览室问题——
- 食品供应问题——
- 三峡大坝问题——
- 睡眠理发师问题——
- 第二类读者写者问题——
- 复杂的消息缓冲问题——
- 考场问题——
- 狒狒过峡谷问题——
- 银行柜员问题——
- 小和尚取水问题——
- 利用信号量解决资源管理问题——
- 睡眠的圣诞老人——

同步问题讲解要求：

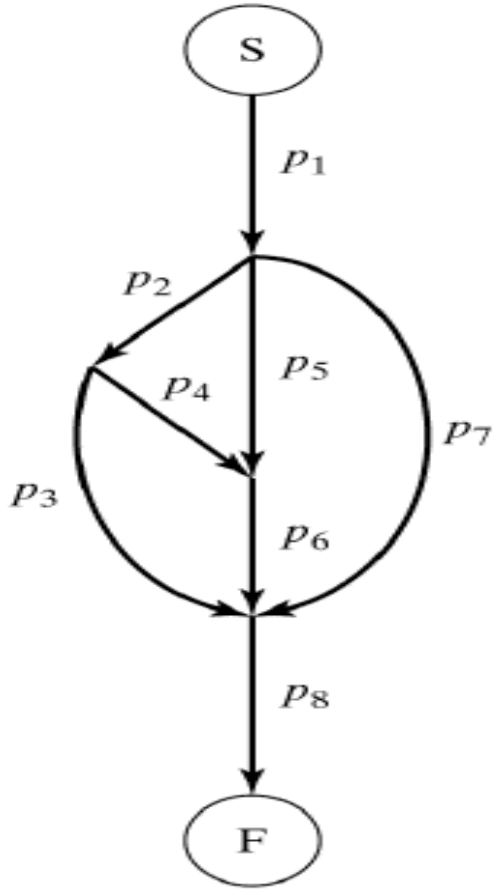
1. 描述问题出现的场景；
2. 描述问题的关键点（同步互斥关系）；
3. 给出问题的解法（代码或伪代码）并说明；
4. 指出哪里容易出现死锁等问题。

制作ppt，发送邮件到教师信箱

纯同步例子



Series/parallel



General precedence

另类P、V操作问题

有一个系统，定义P、V操作如下：

```
P(s):
    s.count --;
    if s<0 then
       将本进程插入相应队列末尾等待;
V(s):
    s.count ++;
    if s<=0 then
       从相应等待队列队尾唤醒一个进程，将其插入就绪队列;
```

思考并回答：

a. 这样定义P、V操作是否有问题?

b. 用这样的P、V操作实现N个进程竞争使用某一共享变量的互斥机制。

c. 对于b的解法，有无效率更高的方法。如有，试问降低了多少复杂性?

睡眠理发师问题

- 理发店里有一位理发师，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，则理发师便在理发椅上睡觉
- 当一个顾客到来时，他必须先唤醒理发师
- 如果顾客到来时理发师正在理发，则如果有空椅子，可坐下来等；否则离开

- 试用P、V操作解决睡眠理发师问题

思考：N个理发师的解决方案

复杂的消息缓冲问题（*）

- 消息缓冲区为k个，有m个发送进程，n个接收进程，每个接收进程对发送来的消息都必须取一次。试用P、V操作解决发送进程和接受进程之间的正确通信问题。

第二类读者写者问题（写者优先）(*)

- 试用信号量及P、V操作解决写者优先问题，要求：

a. 多个读者可以同时进行读；

b. 写者必须互斥(只允许一个写者写，也不能读者写者同时进行)；

c. 写者优先于读者（一旦有写者，则后续读者必须等待，唤醒时优先考虑写者）。

食品供货问题

- 某商店有两种食品A和B， 最大数量各为m个。该商店将A、B两种食品搭配出售，每次各取一个。为避免食品变质，遵循先到食品先出售的原则。有两个食品公司分别不断地供应A、B两种食品(每次一个)。为保证正常销售，当某种食品的数量比另一种的数量超过k(k<m)个时，暂停对数量大的食品进货。试用P、V操作解决上述问题中的同步和互斥关系。

三峡大坝船闸调度问题

- 由于水面高度不同，有160~175米的落差，所以三峡大坝有五级船闸，T_1~T_5。由上游驶来的船需经由各级船闸到下游；由下游驶来的船需经由各级船闸到上游。假设只能允许单方向通行（此假设与实际情况不符，实际为双向各五级船闸）。 试用P、V操作正确解决三峡大坝船闸调度问题。

狒狒过峡谷问题（教材第6章习题）

- 一个主修人类学、辅修计算机科学的学生参加了一个研究课题，调查是否可以教会非洲狒狒理解死锁。他找到一处很深的峡谷，在上边固定了一根横跨峡谷的绳索，这样狒狒就可以攀住绳索越过峡谷。同一时刻，只要朝着相同的方向就可以有几只狒狒通过。但如果向东和向西的狒狒同时攀在绳索上那么会产生死锁（狒狒会被卡在中间），由于它们无法在绳索上从另一只的背上翻过去。如果一只狒狒想越过峡谷，它必须看当前是否有别的狒狒正在逆向通行。利用信号量编写一个避免死锁的程序来解决该问题。不考虑连续东行的狒狒会使得西行的狒狒无限制地等待的情况。

- 重复上一个习题，但此次要避免饥饿。当一只想向东去的狒狒来到绳索跟前，但发现有别的狒狒正在向西越过峡谷时，它会一直等到绳索可用为止。但在至少有一只狒狒向东越过峡谷之前，不允许再有狒狒开始从东向西过峡谷。

考场问题

- 把学生和监考老师都看作进程，学生有N人，教师1人。考场门口每次只能进出一个人，进考场原则是先来先进。当N个学生都进入考场后，教师才能发卷子。学生交卷后可以离开考场，教师要等收上来全部卷子并封装卷子后才能离开考场。

阅览室问题

- 有一个阅览室，共有50个座位，读者进入时必须先在一张登记表上登记，该表为每一座位列一表目，包括座号和读者姓名等，读者离开时要消掉登记的信息，试问：

(1)为描述读者的动作，应编写几个程序，设置几个进程?

(2)试用PV操作描述读者进程之间的关系。

银行柜员问题

- 某银行有人民币储蓄业务，由n个柜员负责。每个顾客进入银行后先取一个号，并且等着叫号。当一个柜台人员空闲下来，就叫下一个号。试用P，V操作正确编写柜台人员和顾客进程的程序。

小和尚取水问题

- 某寺庙，有小和尚、老和尚若干，有一水缸，由小和尚提入水缸供老和尚饮用，水缸可容10桶水，水取自同一井中。水井径窄，每次只能容一个桶取水。水桶总数为3个。每次入缸取水仅为一桶水，且不可同时进行。试给出有关从缸取水、入水的算法描述。

**利用信号量管理共享资源(1/7)

考虑具有如下特征的共享资源：

（1）当使用该资源的进程小于3个时，新申请资源的进程可以立刻获得资源；

（2）当三个资源都被占用后，只有当前使用资源的三个进程都释放资源后，其他申请资源的进程才能够获得资源。

由于需要使用计数器来记录有多少进程正在使用资源和等待资源，而这些计数器自身也需要互斥执行修改动作的共享资源，所以可以采用如下的程序：

(2/7)

```
1 semaphore mutex = 1, block = 0;
2 int active = 0, waiting = 0;
3 boolean must_wait = false;
4
5   P(mutex);
6      if(must_wait) {
7           ++waiting;
8           V(mutex);
9           P(block);
10          P(mutex);
11          --waiting;
12  }
13     ++active;
14     must_wait = active == 3;
15     V(mutex);
16
```

```
17 /* 临界区：对获得的资源进行操作 */
18
19  P(mutex);
20  --active;
21  if(active == 0) {
22    int n;
23    if (waiting < 3) n = waiting;
24    else n = 3;
25    while( n > 0 ) {
26          V(block);
27          --n;
28     }
29       must_wait = false;
30  }
31     V(mutex);
```

利用信号量管理共享资源(3/7)

这个程序看起来没有问题：所有对共享数据的访问均被临界区所保护，进程在临界区中执行时不会自己阻塞，新进程在有三个资源使用者存在时不能使用共享资源，最后一个离开的使用者会唤醒最多3个等待着的进程

- 这个程序仍不正确，解释其出错的位置；
- 假如将第六行的if语句更换为while语句，是否解决了上面的问题？有什么难点仍然存在？

利用信号量管理共享资源(4/7)

现在考虑上一问的正确解法

- 解释这个程序的工作方式，为什么这种工作方式是正确的?
- 这个程序不能完全避免新到达的进程插到已有等待进程前得到资源，但是至少使这种问题的发生减少了。给出一个例子;
- 这个程序是一个使用信号量实现并发问题的通用解法样例，这种解法被称作“I'll Do it for You”（由释放者为申请者修改计数器）模式。解释这种模式。

(5/7)

```
1 semaphore mutex = 1, block = 0;
2 int active = 0, waiting = 0;
3 boolean must_wait = false;
4
5 P(mutex);
6   if(must_wait) {
7     ++waiting;
8     V(mutex);
9     P(block);
10 } else {
11   ++active;
12   must_wait = active == 3;
13   V(mutex);
14 }
15
16 /* 临界区：对获得的资源进行操作 */
17
18   P(mutex);
19   --active;
20   if(active == 0) {
21   int n;
22   if (waiting < 3) n = waiting;
23   else n = 3;
24   waiting -= n;
25   active = n;
26   while( n > 0 ) {
27     V(block);
28     --n;
29   }
30   must_wait = active == 3;
31 }
32   V(mutex);
```

利用信号量管理共享资源(6/7)

现在考虑上一问的另一个正确解法

- 解释这个程序的工作方式，为什么这种工作方式是正确的?
- 这个方法在可以同时唤醒进程个数上是否和上一题的解法有所不同？为什么?
- 这个程序是一个使用信号量实现并发问题的通用解法样例，这种解法被称作“Pass the Baton”（接力棒传递）模式。解释这种模式。

(7/7)

```
1 semaphore mutex = 1, block = 0;
2 int active = 0, waiting = 0;
3 boolean must_wait = false;
4
5 P(mutex);
6 if(must_wait) {
7       ++waiting;
8       V(mutex);
9       P(block);
10      --waiting;
11 }
12 ++active;
13 must_wait = active == 3;
14 if(waiting > 0 && !must_wait)
15      V(block);
16
17 else   V(mutex);
18
```

```
19 /* 临界区：对获得的资源进行操作
*/
20
21   P(mutex);
22   --active;
23   if(active == 0)
24        must_wait = false;
25       if(waiting > 0
&& !must_wait)
26    V(block);
27
28 else V(mutex);
```

睡眠的圣诞老人

- 睡在北极商店中的6名圣诞老人只能被下述情形唤醒：(1) 所有9头驯鹿都从南太平洋度假归来。(2) 有些小精灵在制作玩具时遇到了麻烦；为了让圣诞老人们多休息一会儿，只能在3个小精灵遇到麻烦时才能叫醒圣诞老人。在这3个小精灵解决它们的问题时，其他想要找圣诞老人的小精灵只能等这3个小精灵返回。如果圣诞老人醒来后发现3个小精灵及最后一头从热带度假归来的驯鹿在店门口等着，那么圣诞老人就决定让这些小精灵等到圣诞节以后，因为准备雪橇更加重要（假设驯鹿不想离开热带，因此它们要在那里待到最后可能的时刻）。最后的驯鹿一定要赶回来找圣诞老人，在套上雪橇之前，驯鹿会在温暖的棚子里等着。用信号量及PV操作解决上述问题。

Thanks

The End