西安交通大学
软件学院

操作系统原理

Operating System Principle

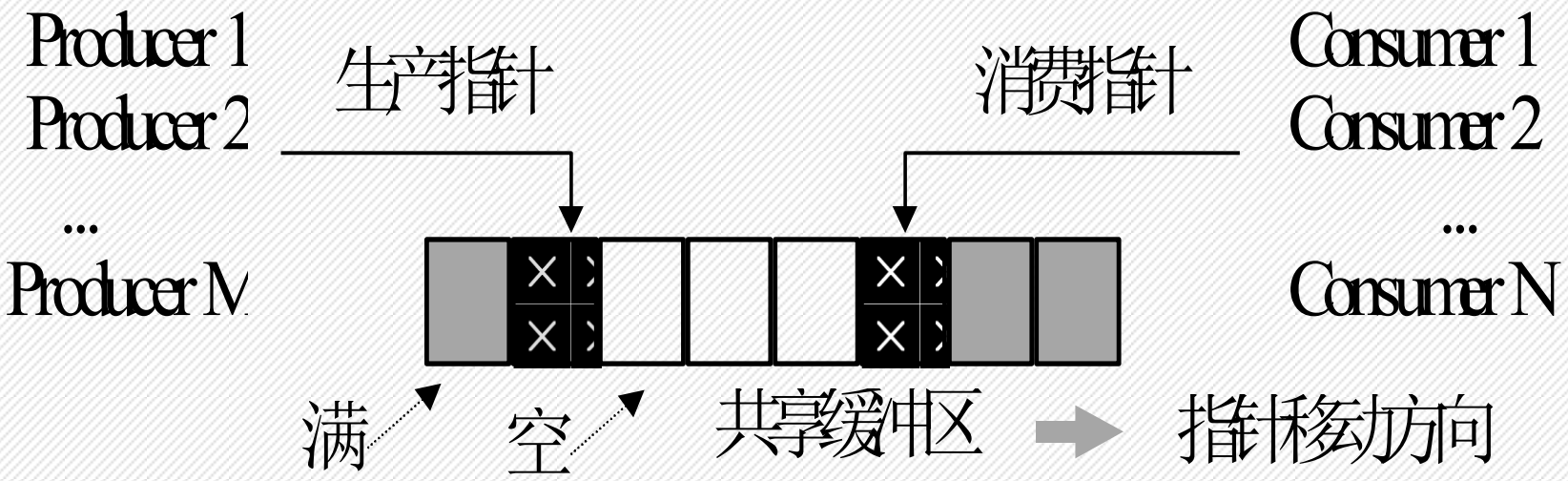
田丽华

6-1 进程同步

生产者消费者问题

采用共享内存解决生产者消费者问题时，N个缓冲区最多只能用N-1个。

如何解决?


Producer 1
Producer 2
...
Producer M
生产指针
消费指针
Consumer 1
Consumer 2
...
Consumer N
满
空
共享缓冲区
指针移动方向

Background
背景

```
// producer calls this method
public void enter(Object item) {
        while (count == BUFFER_SIZE) ; // do nothing
        // add an item to the buffer
        buffer[in] = item;
        in = (in + 1) % BUFFER_SIZE;
        count++;
}
```

Background
背景

```
// consumer calls this method
   public Object remove() {
      Object item;
            while (count == 0) ; // do nothing
             // remove an item from the buffer
              item = buffer[out];
            out = (out + 1) % BUFFER_SIZE;
            count--;
            return item;
   }
```

Background
背景

Count++:

Register1=count;

Register1=register1+1;

Count=register1

Count--:

Register2=count;

Register2=register2-1;

Count=register2

Background
背景

Consider this execution interleaving with "count = 5"initially:

- S0: producer execute register1 = count {register1 = 5}
- S1: producer execute register1 = register1+ 1 {register1 = 6}
- S2: consumer execute register2 = count {register2 = 5}
- S3: consumer execute register2 = register2-1 {register2 = 4}
- S4: producer execute count = register1 {count = 6 }
- S5: consumer execute count = register2 {count = 4}

Background
背景

Concurrent access to shared data may result in data inconsistency（对共享数据的并发访问可能导致数据的不一致性）.

Maintaining data consistency requires mechanisms to ensure the orderly execution of cooperating processes（要保持数据的一致性，就需要一种保证并发进程的正确执行顺序的机制）.

Shared-memory solution to bounded-butter problem (Chapter 4) has a race condition on the class data *count*（解决有界缓冲区问题的共享内存方法在类数据count 上存在竞争条件）

race condition竞争条件

若干个并发的进程(线程)都可以访问和操纵同一个共享数据,从而执行结果就取决于并发进程对这个数据的访问次序.

为了保证数据的一致性,需要有同步机制来保证多个进程对共享数据的互斥访问.

Background
背景

进程类型

协作进程
独立进程

进程间资源访问冲突

共享变量的修改冲突
操作顺序冲突

进程间的制约关系

间接制约：有些资源需要互斥使用，因此各进程进行竞争--独占分配到的部分或全部共享资源，进程的这种关系为进程的互斥
直接制约：进行协作--具体说，一个进程运行到某一点时要求另一伙伴进程为它提供消息，在未获得消息之前，该进程处于等待状态，获得消息后被唤醒进入就绪态.进程的这种关系为进程的同步

Background
背景

一个飞机订票系统，两个终端，运行T1、T2进程

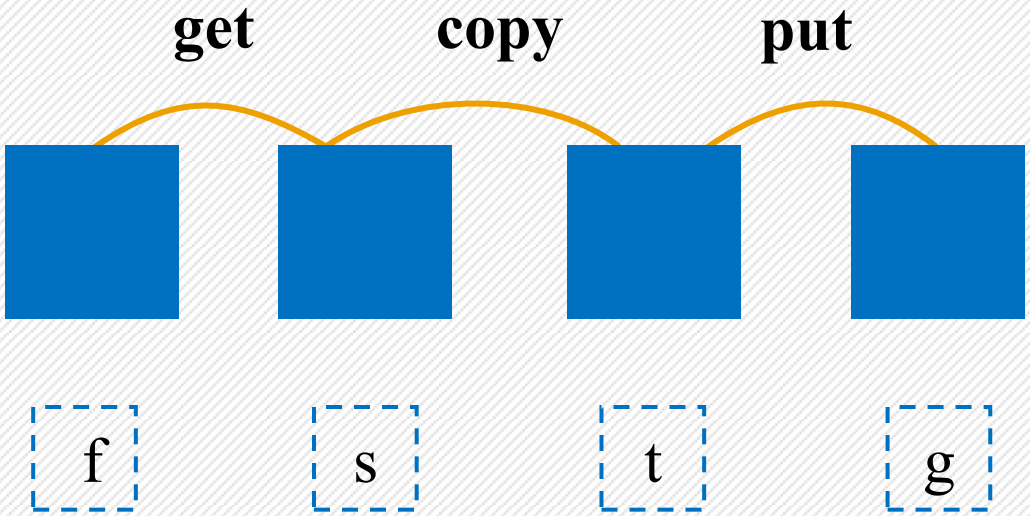
```
T1:
…
Read(x);
if  x>=1 then
    x:=x-1;
write(x);
…
```

```
T2:
…
Read(x);
if  x>=1 then
    x:=x-1;
write(x);
…
```

x = 1

共享变量的修改冲突

Background
背景



有3个进程

get, copy和put，它们对4个存储区域f、s、t和g进行操作。

操作顺序冲突

Interactions between processes
进程间的交互关系

- 互斥，指多个进程不能同时使用同一个资源；
- 同步，进程之间的协作；
- 死锁，指多个进程互不相让，都得不到足够的资源；