西安交通大学

软件学院

操作系统原理

Operating System Principle

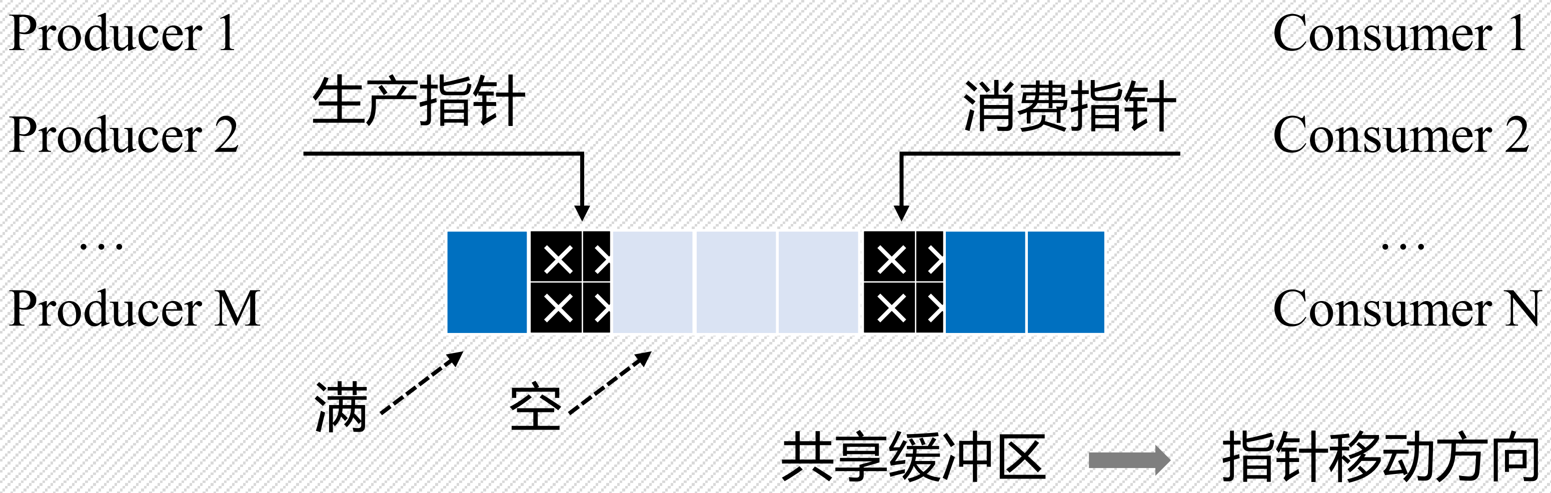
田丽华

6-5 有限缓冲区问题

the producer-consumer problem

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 producer-consumer problem

生产者 - 消费者问题

采用信号量机制：

- full是"满"数目，初值为0，empty是"空"数目，初值为N。实际上，full + empty == N
- mutex用于访问缓冲区时的互斥，初值是1
- 每个进程中各个P操作的次序是重要的：先检查资源数目，再检查是否互斥——否则可能死锁

the producer-consumer problem
生产者 - 消费者问题

Producer

```
P(empty);
P(mutex);      //进入区
One unit → buffer;
V(mutex);
V(full);       //退出区
```

Consumer

```
P(full);
P(mutex);      //进入区
One unit ← buffer;
V(mutex);
V(empty);      //退出区
```

Bounded-Buffer Problem

```
public class BoundedBuffer  {
      public BoundedBuffer() { /* see next slides */  }
      public void enter() { /* see next slides */  }
      public Object remove() { /* see next slides */  }

    private static final int   BUFFER_SIZE = 2;
      private Semaphore mutex;
      private Semaphore empty;
      private Semaphore full;
      private int in, out;
      private Object[] buffer;
}
```

the producer-consumer problem

生产者 - 消费者问题

Bounded Buffer Constructor

```
public BoundedBuffer() {
        // buffer is initially empty
        count = 0;
        in = 0;
        out = 0;
        buffer = new Object[BUFFER_SIZE];
        mutex = new Semaphore(1);
        empty = new
Semaphore(BUFFER_SIZE);
        full = new Semaphore(0);
    }
```

the producer-consumer problem

生产者 - 消费者问题

enter() Method

```
public void enter(Object item) {
    empty.P();
    mutex.P();

    // add an item to the buffer
    buffer[in] = item;
    in = (in + 1) % BUFFER_SIZE;
    mutex.V();
    full.V();
}
```

the producer-consumer problem

生产者 - 消费者问题

remove() Method

```
public Object remove() {
      full.P();
      mutex.P();

      // remove an item from the buffer
      Object item = buffer[out];
      out = (out + 1) % BUFFER_SIZE;

      mutex.V();
      empty.V();

      return item;
   }
```