操作系统原理及Linux内核

西安邮电大学

生产者——消费者问题

单生产者--单消费者问题

单缓冲区

系统中有一个生产者进程、一个消费者进程和一个一次只能放1个产品的缓冲区。生产者进程重复的生产产品并放入到缓冲区中；每当缓冲区中有产品时，消费者进程从缓冲区中取产品进行消费。

信号量：space，存储位置，初值1； prod，产品，初值0

单生产者--单消费者问题(单缓冲区)

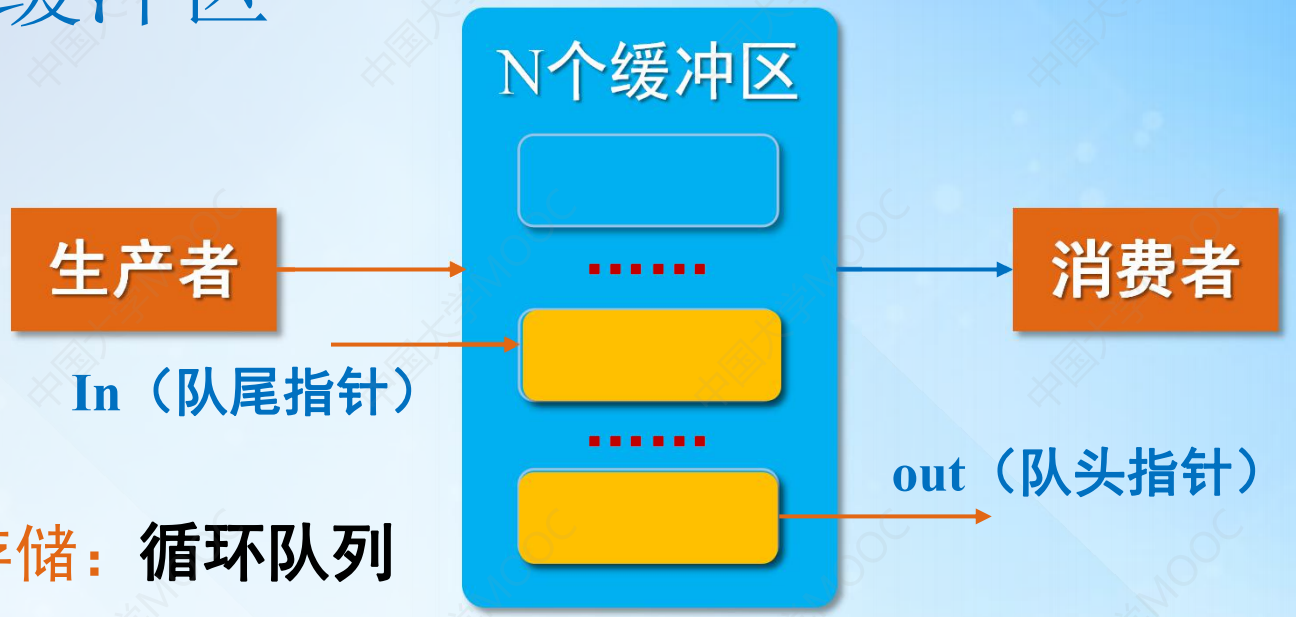
```
#include <semaphore.h>
sem_t space, prod;
void * producer(void *p){
    while(1){
        sem_wait(&space);
        printf("Put a product\n");
        sem_post(&prod);
    }
    return NULL;
}
```

```
void * consumer(void *p){
    while(1){
        sem_wait(&prod);
        printf("Gets a product\n");
        sem_post(&space);
    }
    return NULL;
}
```

```
    pthread_join(tid[0], NULL);
    return 0;
}
```

单生产者--单消费者问题

多缓冲区



产品存储：循环队列

信号量：space，存储位置，初值N；prod, 产品，初值0

单生产者--单消费者问题(多缓冲区)

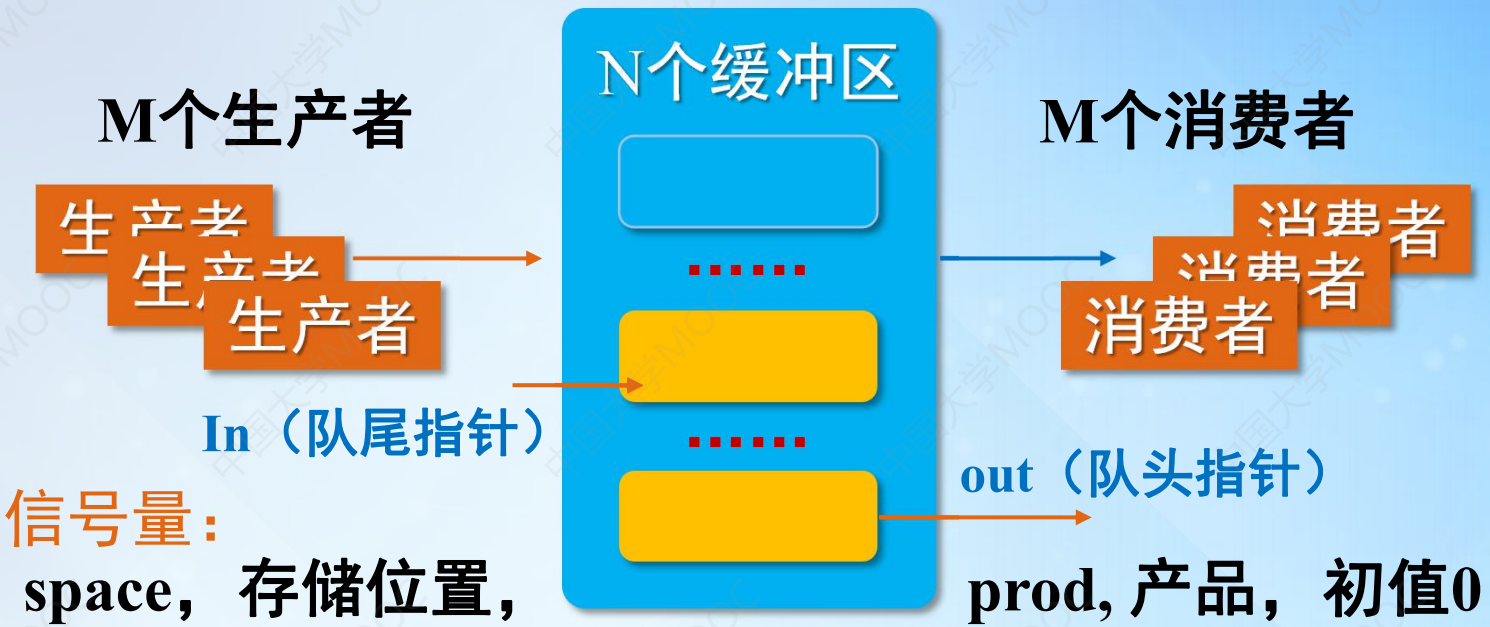
```
#define N 10
```

```
void * producer(void *p){
    while(1){
        sem_wait(&space);
        printf("Put a product into
            Buffer[%d]!\n", in);
        in = (in +1)%N;
        sem_post(&prod);
    }
    return NULL;
}
```

```
void * consumer(void *p){
    while(1){
        sem_wait(&prod);
        printf ("Get a product from
            Buffer[%d]!\n", out);
        out = (out +1)%N;
        sem_post(&space);
    }
    return NULL;
}
```

}

多生产者--多消费者问题



信号量：

space，存储位置，初值N；

prod，产品，初值0

buf，缓冲区，初值1；

多生产者--多消费者问题

```
void * producer(void *p){
    while(1){
        sem_wait(&space);
        sem_wait(&buf);
        printf("Put a product into
            Buffer[%d]!\n", in);
        in = (in +1)%N;
        sem_post(&prod);
        sem_post(&buf);
    }
    return NULL;
}
```

```
void * consumer(void *p){
    while(1){
        sem_wait(&buf);
        sem_wait(&prod);
        printf ("Get a product from
            Buffer[%d]!\n", out);
        out = (out +1)%N;
        sem_post(&space);
        sem_post(&buf);
    }
    return NULL;
}
```

可否交换

多生产者--多消费者问题(ADD)

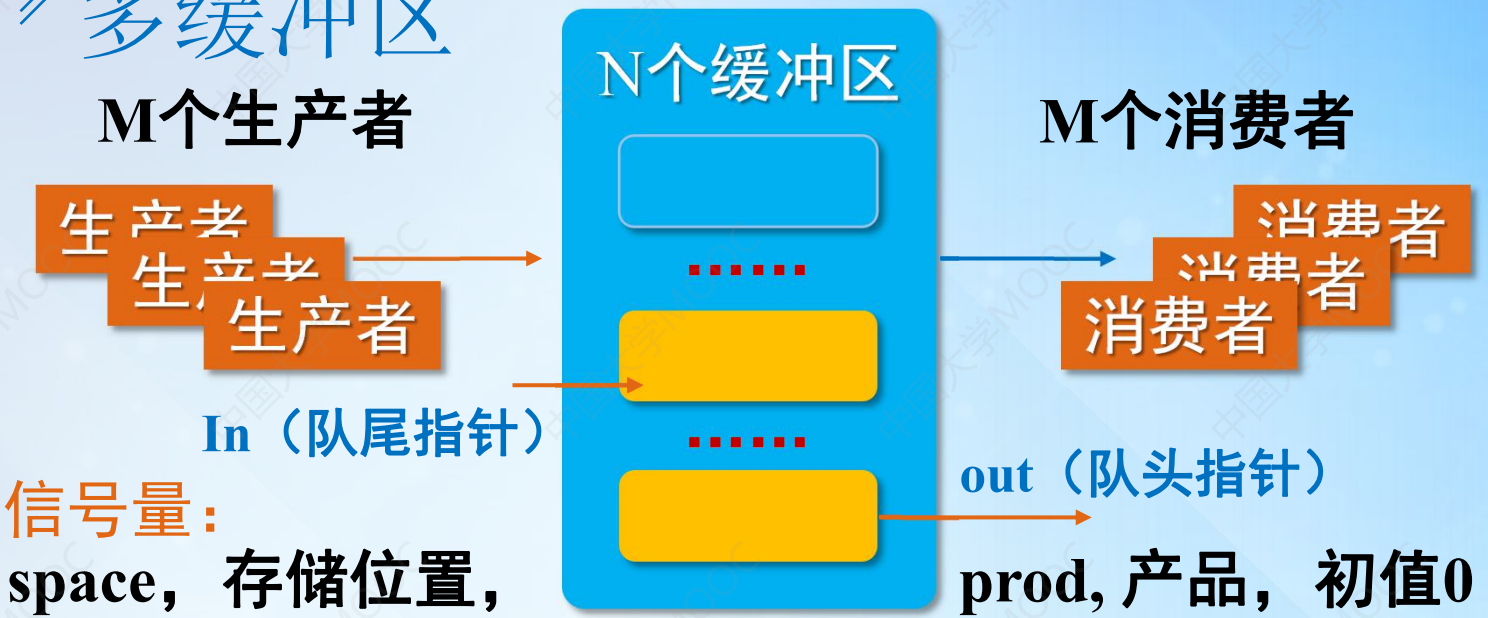
```
#define M 8
void * producer(void *p){
    while(1){
        Swait(space, buf);
        printf("Put a product into
            Buffer[%d]!\n", in);
        in = (in +1)%N;
        Ssignal(prod, buf);
    }
    return NULL;
}
```

```
void * consumer(void *p){
    while(1){
        Swait(prod, buf);
        printf ("Get a product from
            Buffer[%d]!\n", out);
        out = (out +1)%N;
        Ssignal(space, buf);
    }
    return NULL;
}
```

```
    ……
}
```

多生产者--多消费者问题（优化）

多缓冲区



信号量：

space，存储位置，初值N；

buf，缓冲区，初值1；

prod，产品，初值0

sin，队尾，初值1；

sout，队头，初值1

多生产者--多消费者问题（优化）

```
void * producer(void *p){
    while(1){
        sem_wait(&space);
        sem_wait(&sin);
        printf("Put a product into
            Buffer[%d]!\n", in);
        in = (in +1)%N;
        sem_post(&prod);
        sem_post(&sin);
    }
    return NULL;
}
}
```

```
void * consumer(void *p){
    while(1){
        sem_wait(&prod);
        sem_wait(&sout);
        printf ("Get a product from
            Buffer[%d]!\n", out);
        out = (out +1)%N;
        sem_post(&space);
        sem_post(&sout);
    }
    return NULL;
}
```

可否交换？

小结

生产者—消费者问题

- 单-单(多)
  - 生—消 同步
  - space、prod
- 单-单(多)
  - 生—消 同步
  - space、prod
  - 循环队列
- 多-多(多)
  - 生—消 同步
  - 注意死锁
  - 生—生、消—消互斥
- 解决策略
  - 寻找临界资源
  - 使用前申请
  - 定义信号量
  - 使用后释放

字体：中文：思源黑体≥24
英文：新罗马 ≥24

配色

层级  文件管理

特殊字体双击安装