

操作系统

Operating Systems

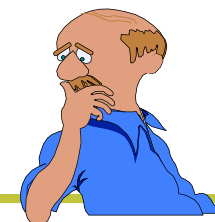
L16 进程同步与信号量

Processes Synchronization and Semaphore

授课教师：李治军

lizhijun_os@hit.edu.cn
综合楼411室

进程合作：多进程共同完成一个任务



实例1:

司机

```
while (true){
```

等待



启动车辆;

正常运行;

到站停车;



售票员

```
while (true){
```

关门;



售票;

等待

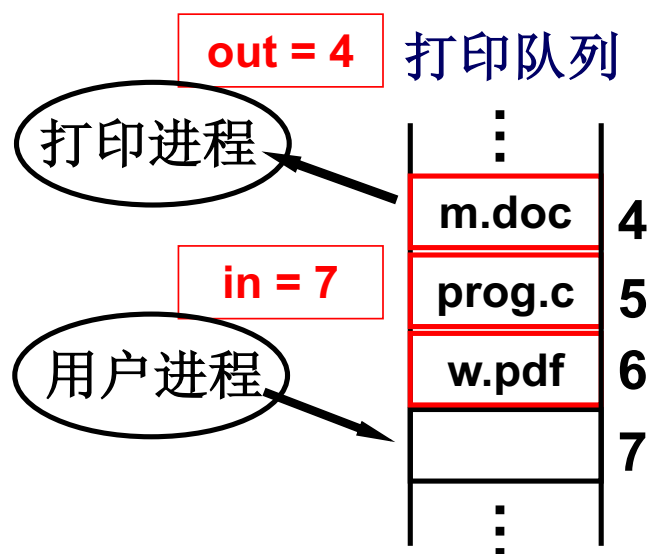


开门; }

进程直接相互发信号
来控制各个进程的前进

实例2: 文档打印

问题：如果进程之间完全不知道对方的存在，则可能会产生怎样的错误？
生活常识是怎么办的？



从纸上到实际：生产者-消费者实例

共享数据

```
#define BUFFER_SIZE 10
typedef struct { . . . } item;
item buffer[BUFFER_SIZE];
int in = out = counter = 0;
```

注意：这些都是用
户态程序！

生产者进程

```
while (true) {
    while(counter == BUFFER_SIZE)
        ;
    buffer[in] = item;
    in = (in + 1) % BUFFER_SIZE;
    counter++;
}
```

消费者进程

```
while (true) {
    while(counter == 0)
        ;
    item = buffer[out];
    out = (out + 1) % BUFFER_SIZE;
    counter--;
}
```



找到哪些地方要停，什么时候再走？

- 需要让“进程走走停停”来保证多进程合作的合理有序

这就是进程同步

生产者进程

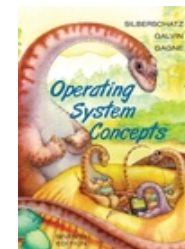
```
while (true) {  
    while(counter == BUFFER_SIZE)  
        缓存区满，生产者要停;  
    buffer[in] = item;  
    in = (in + 1) % BUFFER_SIZE;  
    counter++;  
}
```

发信号让消费者再走...

消费者进程

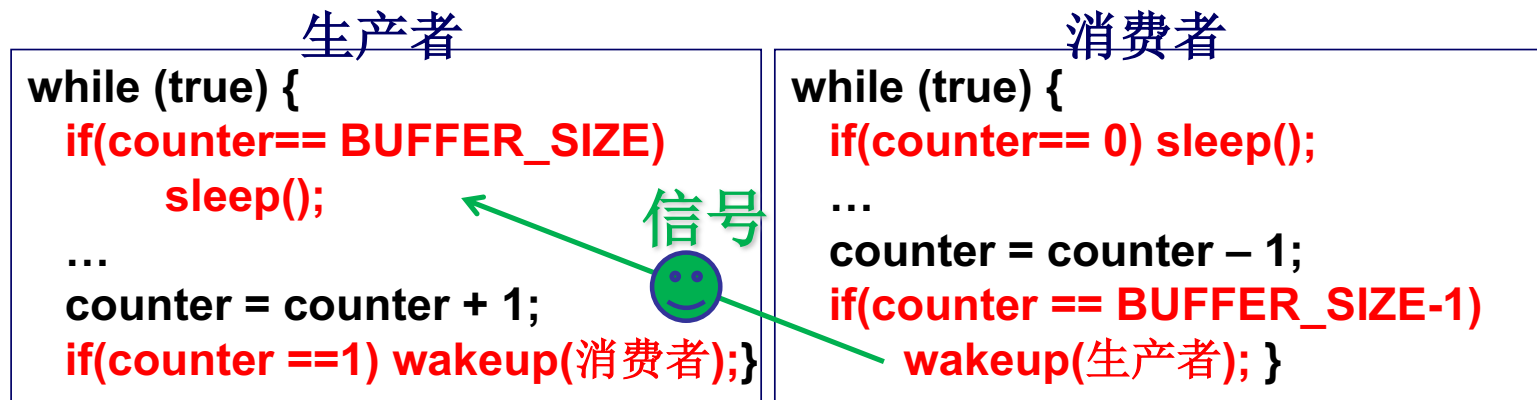
```
while (true) {  
    while(counter == 0)  
        缓存区空，消费者要停;  
    item = buffer[out];  
    out = (out + 1) % BUFFER_SIZE;  
    counter--;  
}
```

发信号让生产者再走...



只有信号不能够解决有多个生产者阻塞的情况

只发信号还不能解决全部问题



- (1) 缓冲区满以后生产者 P_1 生产一个item放入，会sleep
- (2) 又一个生产者 P_2 生产一个item放入，会sleep
- (3) 消费者C执行1次循环，`counter==BUFFER_SIZE-1`，发信号给 P_1 ， P_1 wakeup
- (4) 消费者C再执行1次循环，`counter==BUFFER_SIZE-2`， P_2 不能被唤醒

问题：怎么办？

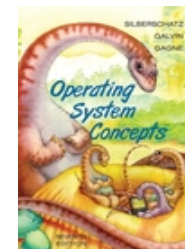


从信号到信号量

信号量表示现在拥有的资源的个数

- 不只是等待信号、发信号？ 对应睡眠和唤醒！
- 还应该能记录一些信息
 - 能记录有“2个进程等待”就可以了...
 - (1) 缓冲区满， P_1 执行， P_1 sleep，记录下1个进程等待
 - (2) P_2 执行， P_2 sleep，记录下2个进程等待
 - (3) C执行1次循环，发现2个进程等待，wakeup 1个
 - (4) C再执行1次循环，发现?个进程等待，再？
 - (5) C再执行1次循环，怎么办？此时再来 P_3 怎么办？

什么是信号量？ 记录一些信息(量)，并根据这个信息决定睡眠还是唤醒(信号)。



多个进程根据信号量的值来决定自己是否需要
阻塞或者唤起别的进程!

信号量开始工作...

初始时 **sem = ?**

■ (1) 缓冲区满, P_1 执行, P_1 sleep

sem = -1 什么含义? 表示缺一个东西

■ (2) P_2 执行, P_2 sleep

sem = -2 从阻塞队列中取出一个weakeup

■ (3) C执行1次循环, wakeup P_1

sem = -1

■ (4) C再执行1次循环, wakeup P_2

sem = 0

用信号量记录有
多少个可以唤醒

■ (5) C再执行1次循环,

sem = 1 什么含义?

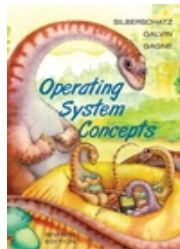
表示现在还有一个缓冲区,
不需要睡眠

■ (6) P_3 执行, P_2 继续执行

sem = 0

■ 总结一下: 这个整数什么时候-1? 什么时候+1?

什么时候睡眠? 什么时候唤醒? 根据信号量来实现进程之间的同步



问题：一种资源的数量是8，这个资源对应的信号量的当前值是2，说明：() **B**

- A. 有2个进程等待这个资源
- B. 有2个资源可以使用
- C. 有6个进程等待这个资源
- D. 有6个资源可以使用



这里的PV操作是需要操作PCB的，所以需要在内核态中执行，
所以需要将PV做成系统调用，供上层使用

什么是信号量？信号量的定义...

- 信号量: 1965年，由荷兰学者Dijkstra提出的一种特殊整型变量，量用来记录，信号用来sleep和wakeup

```
struct semaphore
{
    int value; //记录资源个数
    PCB *queue;
    //记录等待在该信号量上的进程
}
P(semaphore s);    //消费资源
V(semaphore s);    //产生资源
```

```
P(semaphore s)
{
    s.value--;
    if(s.value < 0) {
        sleep(s.queue);
    }
```

每个信号量还需要
记录一个阻塞的队列!

问题：写出V(s)的代码？

```
V(s){
    s.val ++;
    if(s.val <= 0){
        wakeup(s.queue);
    }
}
```

P的名称来源于荷兰语的proberen，即test
V的名称也来源于荷兰语verhogen(increment)



用信号量解生产者-消费者问题

```
int fd = open("buffer.txt");  
write(fd, 0, sizeof(int)); //写in  
write(fd, 0, sizeof(int)); //写out
```

用文件定义共享缓冲区

```
semaphore full = 0;  
semaphore empty = BUFFER_SIZE;  
semaphore mutex = 1;
```

信号量的定义和初始化

```
Producer(item) {  
    P(empty);  
    P(mutex);  
    读入in;将item写入到  
in的位置上;  
    V(mutex);  
    V(full); }
```

```
Consumer() {  
    P(full);  
    P(mutex);  
    读入out;从文件中的out  
位置读出到item;打印item;  
    V(mutex);  
    V(empty); }
```

mutex 互斥信号量
值为1的

