西安交通大学
软件学院

# 操作系统原理

## Operating System Principle

田丽华

# 6-3 信号量

# Semaphore

- 可以用临界区解决互斥问题，它们是平等进程间的一种协商机制
- OS可从进程管理者的角度来处理互斥的问题，信号量就是OS提供的管理公有资源的有效手段。

- 1965年，由荷兰学者Dijkstra提出（所以P、V分别是荷兰语的test(proberen)和increment(verhogen)），是一种卓有成效的进程同步机制。
- 用于保证多个进程在执行次序上的协调关系的相应机制称为进程同步机制。

**信号量代表可用资源实体的数量。**

# Semaphore

- **Semaphore *S* – integer variable**（信号量*S* – 整型变量，代表可用资源实体的数量）
- **can only be accessed via two indivisible (atomic) operations**（除了初始化之外，仅能通过两个不可分割的[原子]操作访问，）

```
P (S):
        while S≤ 0 do no-op;
        S--;

V(S): S++;
```

- 存在忙等——自旋锁:进程在等待锁时自旋

- **Synchronization tool that does not require busy waiting**（一种不需要忙等待的同步工具）.

```
P (S):
        S--;
        if  S< 0 do block;
V(S):
        S++;
        if S<=0 then wakeup;
```

# Semaphore Eliminating Busy-Waiting

## 纪录型信号量

```
semaphore s;
P(s) {
    s.value--;
    if (s.value < 0) {
        add this process to list s.L
        block
    }
}
V(s) {
    s.value++;
    if (s.value <= 0) {
        remove a process P from list
s.L
        wakeup(P);
    }
}
```

```
typedef struct{
    int value;
    struct process *L;
}semaphore
```

# Semaphore

S是与临界区内所使用的公用资源有关的信号量

P(S):表示申请一个资源

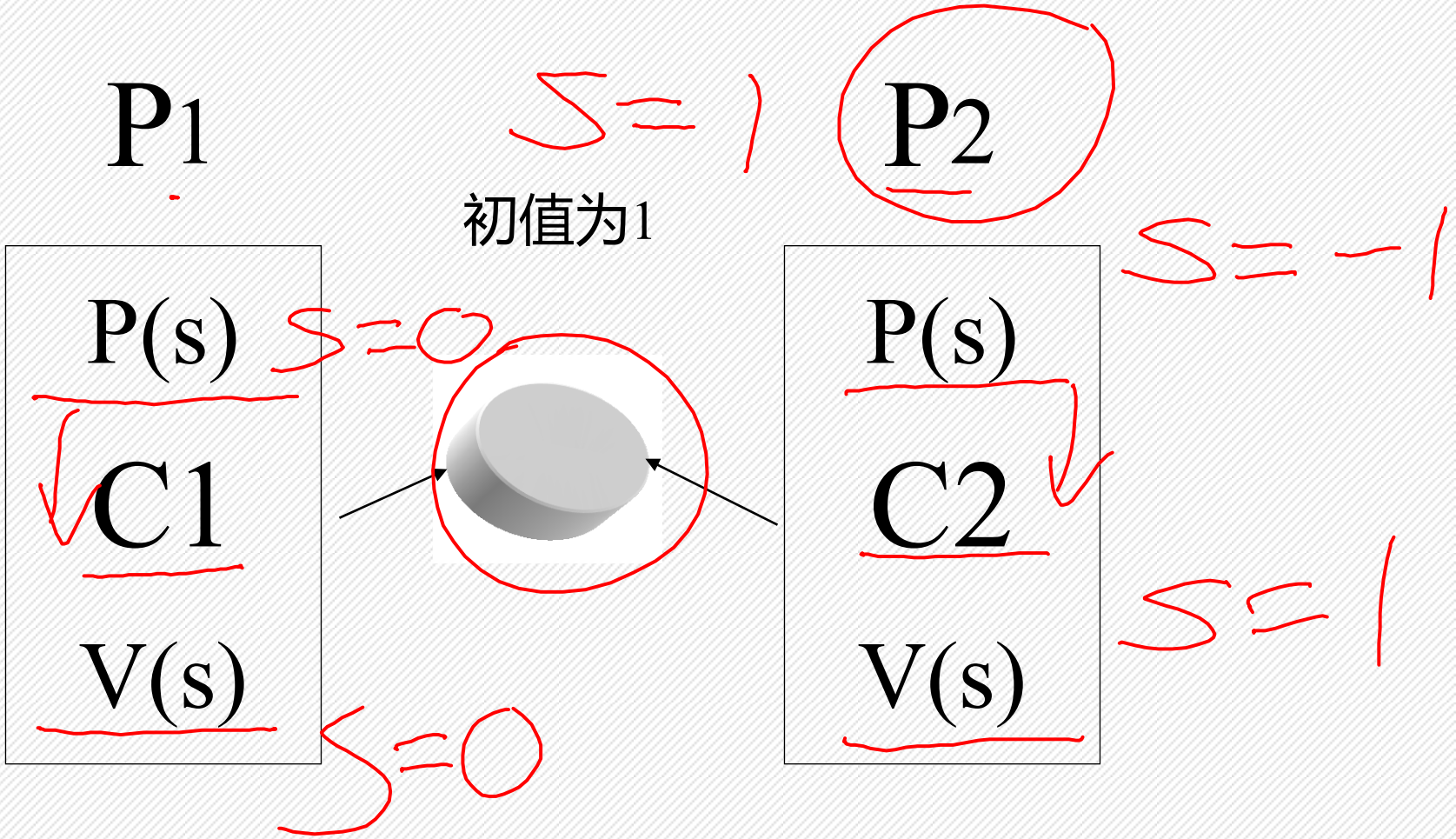
V(S):表示释放一个资源。

- 初始化指定一个非负整数值，表示空闲资源总数

在信号量经典定义下，信号量S的值不可能为负

- S≥0 可供并发进程使用的资源数
- S<0 其绝对值就是正在等待进入临界区的进程数

# 利用信号量来描述互斥关系



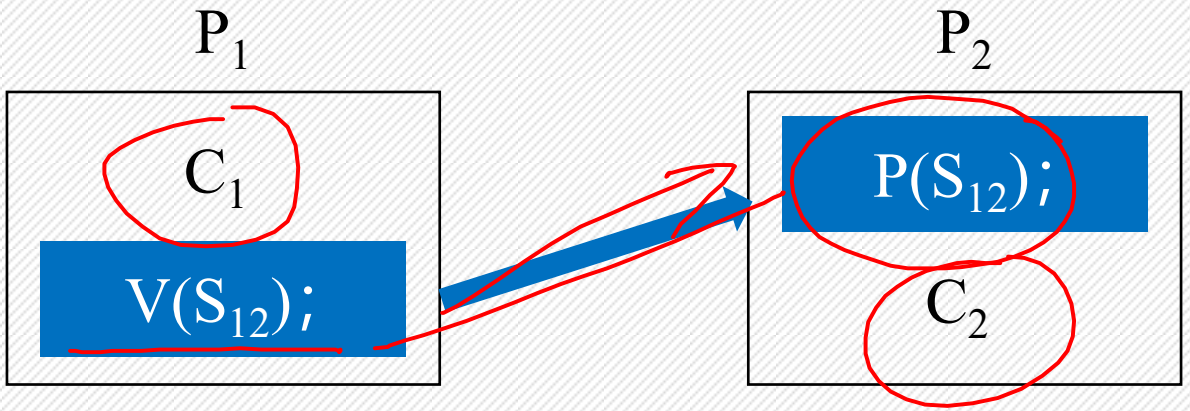
# Semaphore as General Synchronization Tool

利用信号量实现互斥：
为临界资源设置一个互斥信号量，其初值为1；
**Semaphore S; // initialized to 1**

在每个进程中将临界区代码置于P(S)和V(S)原语之间

```
P(S);
    CriticalSection()
V(S);
```

# 利用信号量来描述前趋关系




- 前趋关系：并发执行的进程**P1**和**P2**中，分别有代码**C1**和**C2**，要求**C1**在**C2**开始前完成；
- 为每个前趋关系设置一个同步信号量**S12**，其初值为**0**