sem_init 函数将信号量 sem 初始化为 value。每个信号量在使用前必须初始化。针对我们的目的，中间的参数总是零。程序分别通过调用 sem_wait 和 sem_post 函数来执行 P 和 V 操作。为了简明，我们更喜欢使用下面这些等价的 P 和 V 的包装函数：

```
#include "csapp.h"

void P(sem_t *s);   /* Wrapper function for sem_wait */
void V(sem_t *s);   /* Wrapper function for sem_post */
```

返回：无。

旁注　P 和 V 名字的起源

Edsger Dijkstra(1930—2002)出生于荷兰。名字 P 和 V 来源于荷兰语单词 Proberen(测试)和 Verhogen(增加)。

12.5.3　使用信号量来实现互斥

信号量提供了一种很方便的方法来确保对共享变量的互斥访问。基本思想是将每个共享变量(或者一组相关的共享变量)与一个信号量 s(初始为 1)联系起来，然后用 $P(s)$和 $V(s)$操作将相应的临界区包围起来。

以这种方式来保护共享变量的信号量叫做*二元信号量*(binary semaphore)，因为它的值总是 0 或者 1。以提供互斥为目的的二元信号量常常也称为*互斥锁*(mutex)。在一个互斥锁上执行 P 操作称为对互斥锁*加锁*。类似地，执行 V 操作称为对互斥锁*解锁*。对一个互斥锁加了锁但是还没有解锁的线程称为*占用*这个互斥锁。一个被用作一组可用资源的计数器的信号量被称为*计数信号量*。

图 12-22 中的进度图展示了我们如何利用二元信号量来正确地同步计数器程序示例。每个状态都标出了该状态中信号量 s 的值。关键思想是这种 P 和 V 操作的结合创建了一组

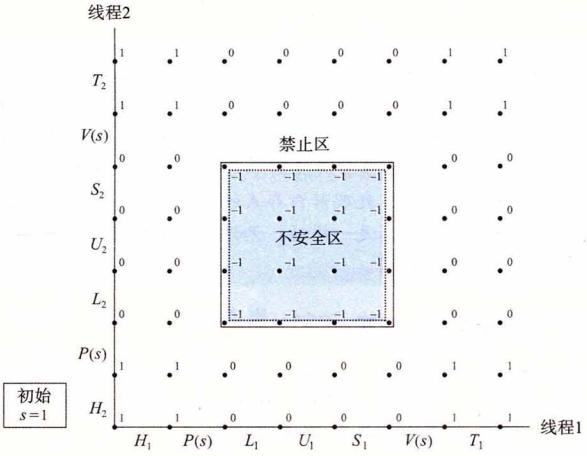

图 12-22　使用信号量来互斥。$s<0$ 的不可行状态定义了一个禁止区，禁止区完全包括了不安全区，阻止了实际可行的轨迹线接触到不安全区