并发编程

金超 章梓立

2021-12-21

Keywords

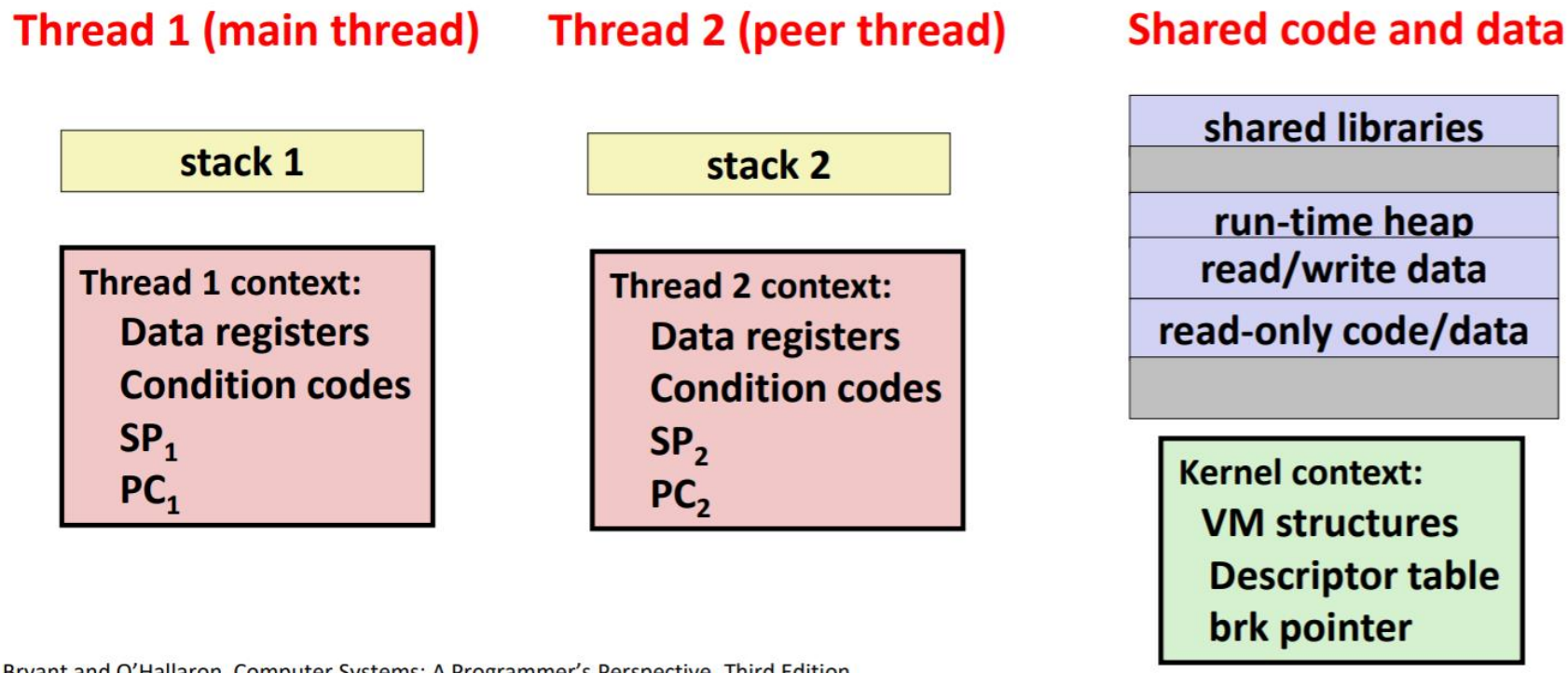
- 线程
  - 并发和并行的区别
  - 并发echoserver的三种实现方式
  - 线程和进程的区别
  - `pthread`函数
- 同步
  - 不安全区
  - 信号量 P V 禁止区
- 同步问题：竞争 死锁 饥饿

进程 vs 线程

- 进程：运行中的程序实例，CPU+内存的状态（进程上下文）
  - 基于进程的并发：地址空间相互独立，难以实现共享，切换开销大
- 线程：“轻量级”进程，某个进程上下文的一组控制流（CPU的状态）
  - 基于线程的并发：共享地址空间等进程上下文，切换开销小
  - 高度的共享引起竞争、饥饿、死锁等问题，在线程模型中这个问题比进程的时候要更多更大。

进程 vs 线程

- 同一进程的不同线程



Posix threads

- #include <pthread.h>
- 线程的大多数接口功能和进程系统调用相似
- 关键区别：所有的线程都是同级关系，因此它们彼此都能互相创建、互相回收

功能	线程	进程	区别
创建	pthread_create	fork	创建者和被创建的关系；线程可以通过参数指定逻辑流的入口，进程需要根据返回值进行设定
获取id	pthread_self	getpid	
终止	pthread_exit/pthread_cancel	exit	任一线程执行exit会终止所有对等线程；线程间提供了直接终止某个对等线程的接口，而进程需要通过信号机制
回收	pthread_join	Wait/waitpid	前者只能等待特定线程终止
分离	pthread_detach		结束时能自动释放资源，不用专门阻塞其它某个线程；如果我们不在乎某个线程的终止状态和返回值，就可以没有负担地分离它

同步

- 竞争
  - 线程共享地址空间
  - 绝大多数操作不是原子性的
  - i++; -> mov(load), add(use), mov(store)
- 死锁
  - 进度图 & 禁止区
- 信号量
  - P，V都是原子操作
  - P(&s) while(s == 0) {wait() } s--; 等待直到可以运行
  - V(&s) s++; 好了
  - 生产者-消费者问题（当缓冲区满/空时，生产者/消费者不能继续）
  - 读者-写者问题（当读者or写者在访问时，写者不能继续；当写者在访问时，读者不能继续）
  - 把复杂问题转化成已经学过的两个问题

线程安全

- 线程不安全函数
  - 不保护共享变量
  - 保持跨越多个调用的状态
  - 返回指向静态变量的指针
  - 调用线程不安全函数
- 可重入函数

题目讲解

完成题目 时间20min

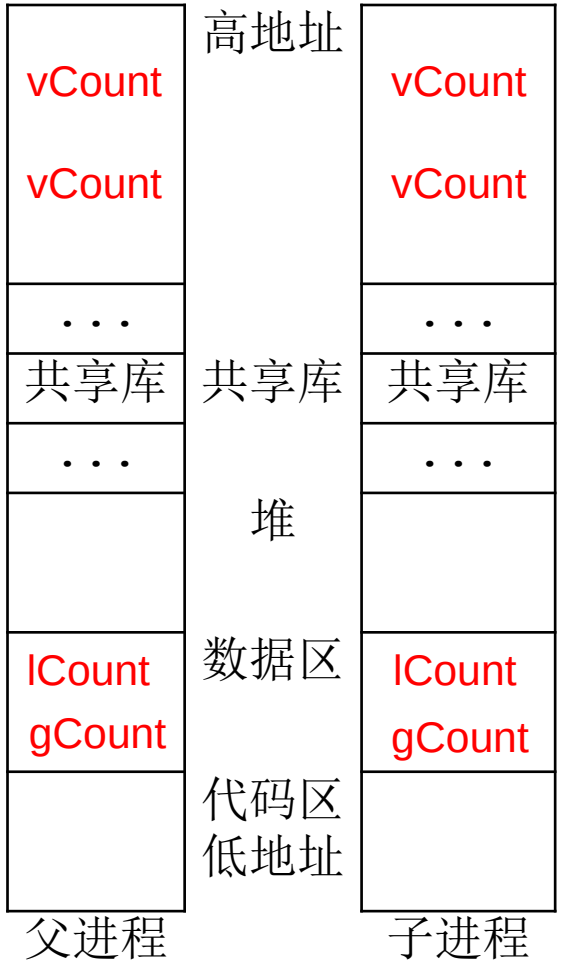
共享变量

栈上的临时变量
线程不共享

局部静态变
位于数据区
线程共享

全局变量
位于
线

```
long gCount = 0;
void *thread(void *vargp) {
    volatile long vCount = *(long *)vargp;
    static long lCount = 0;
    gCount++; vCount++; lCount++;
    printf("%ld\n", gCount+vCount+lCount);
    return NULL;
}
int main() {
    long var; pthread_t tid1, tid2;
    scanf("%ld", &var);
    fork();
    pthread_create(&tid1,NULL,thread,&var);
    pthread_create(&tid2,NULL,thread,&var);
    pthread_join(tid1, NULL);
    pthread_join(tid2, NULL);
    return 0;
}
```



线程竞争

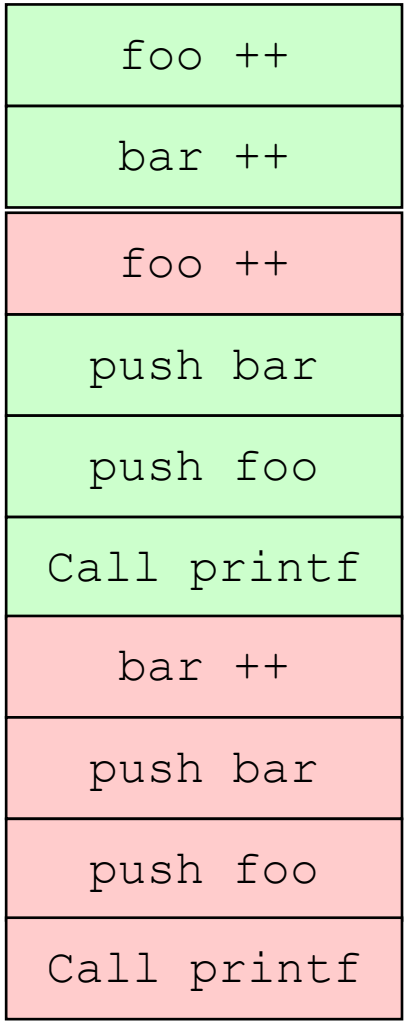
下面的程序会引发竞争。一个可能的输出结果为2 1 2 2。解释输出这一结果的原因。

```
long foo = 0, bar = 0;

void *thread(void *vargp) {
    foo++;
    bar++;
    printf("%ld %ld ", foo, bar);
    fflush(stdout);
    return NULL;
}

int main() {
    pthread_t tid1, tid2;
    pthread_create(&tid1, NULL, thread, NULL);
    pthread_create(&tid2, NULL, thread, NULL);
    pthread_join(tid1, NULL);
    pthread_join(tid2, NULL);
    return 0;
}
```

线程竞争

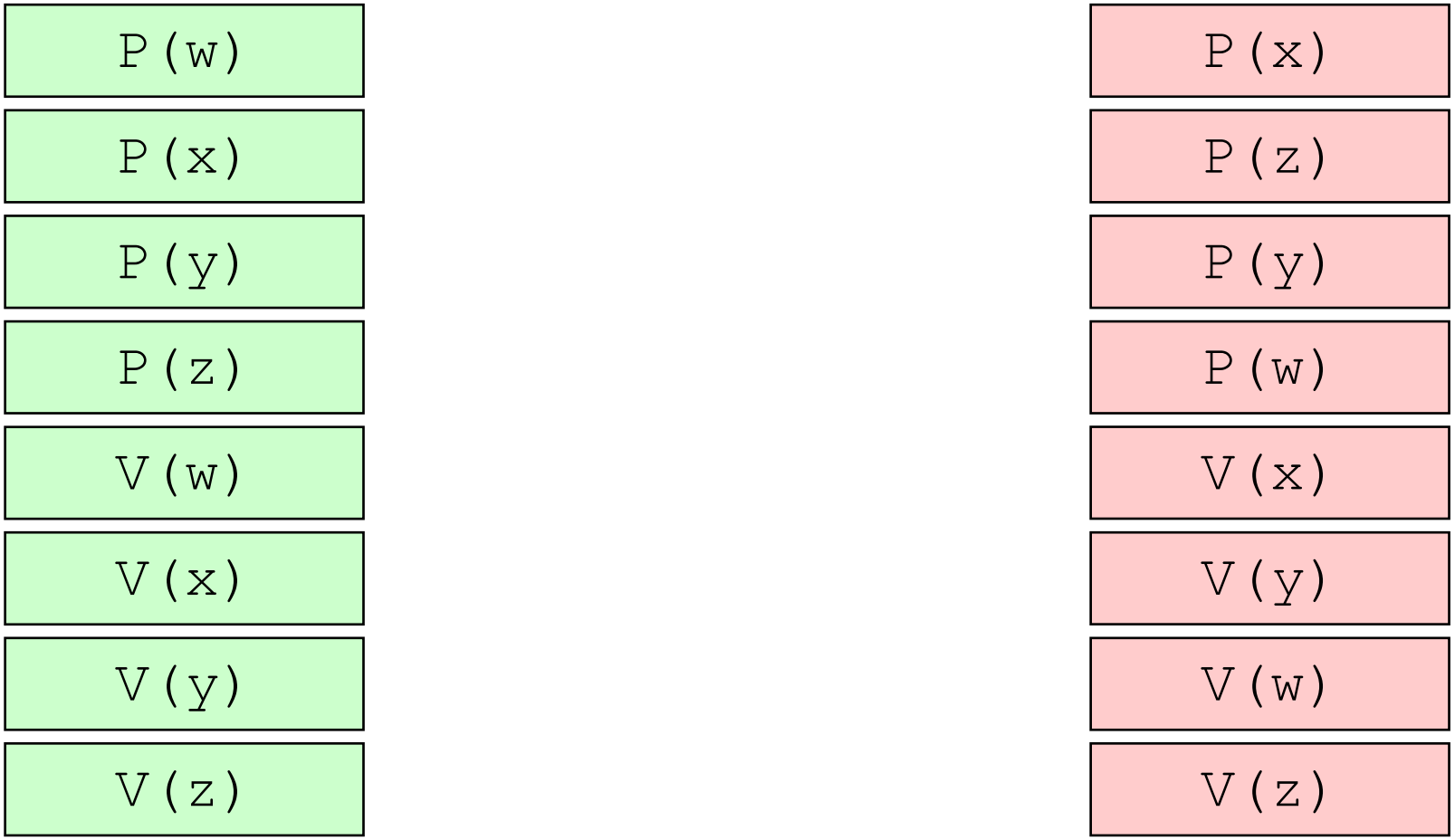

foo ++
bar ++
foo ++
push bar
push foo
Call printf
bar ++
push bar
push foo
Call printf
foo = 1, bar = 1
foo = 2, bar = 1
2 1
foo = 2, bar = 2
2 2

死锁

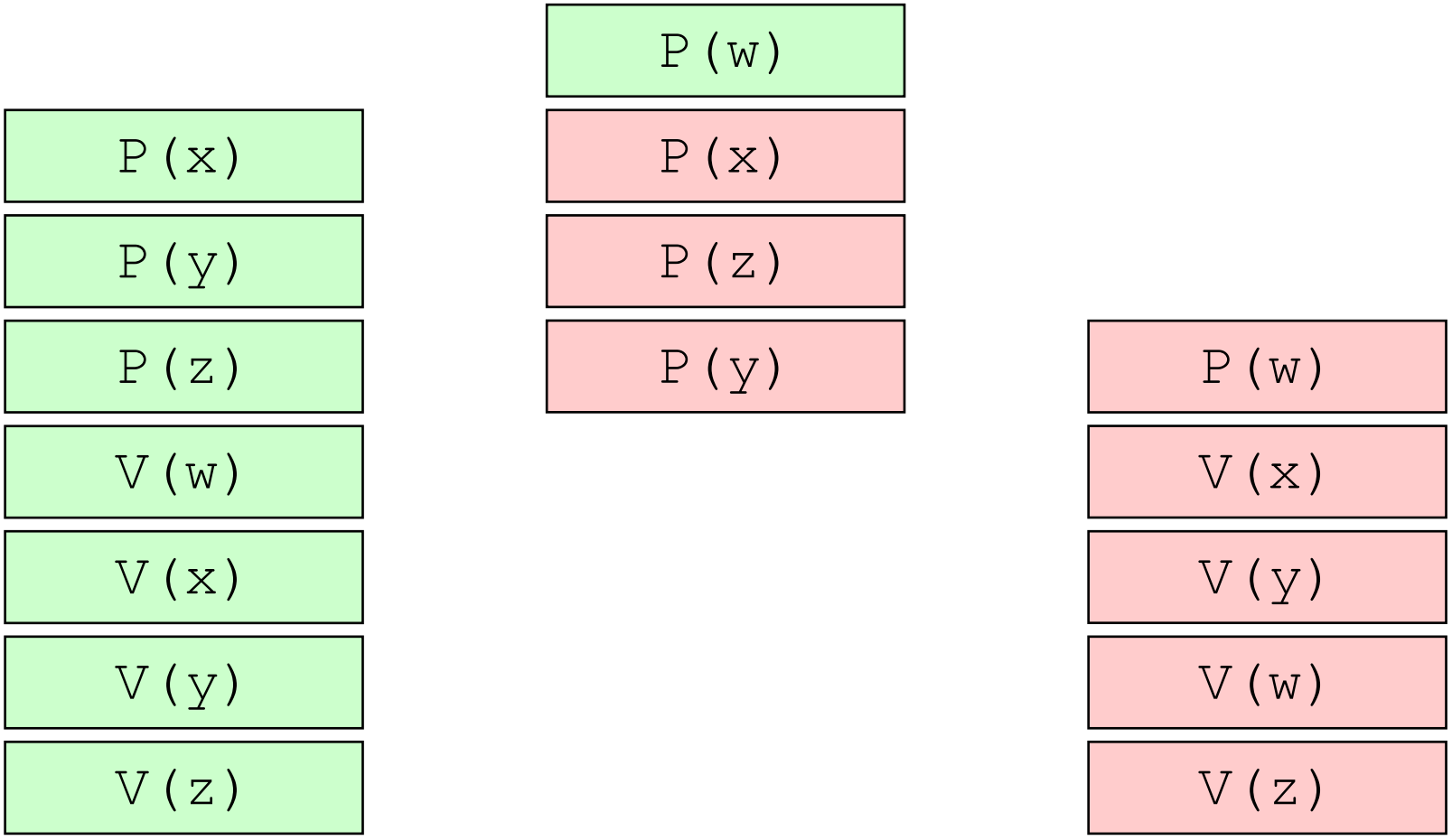
3． 信号量 w,x,y,z 均被初始化为 1。下面的两个线程运行时可能会发生死锁。给出发生死锁的执行顺序。

线程 1	①P(w) ②P(x) ③P(y) ④P(z) ⑤V(w) ⑥V(x) ⑦V(y) ⑧V(z)
线程 2	Ⅰ P(x) Ⅱ P(z) Ⅲ P(y) Ⅳ P(w) Ⅴ V(x) Ⅵ V(y) Ⅶ V(w) Ⅷ V(z)

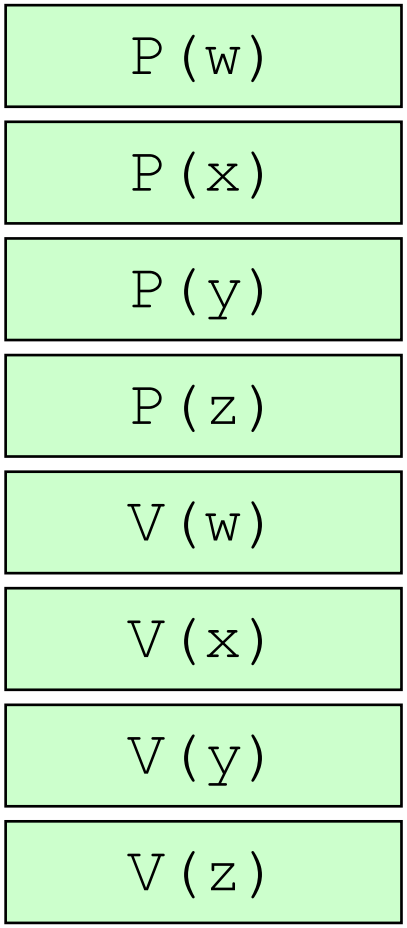
死锁



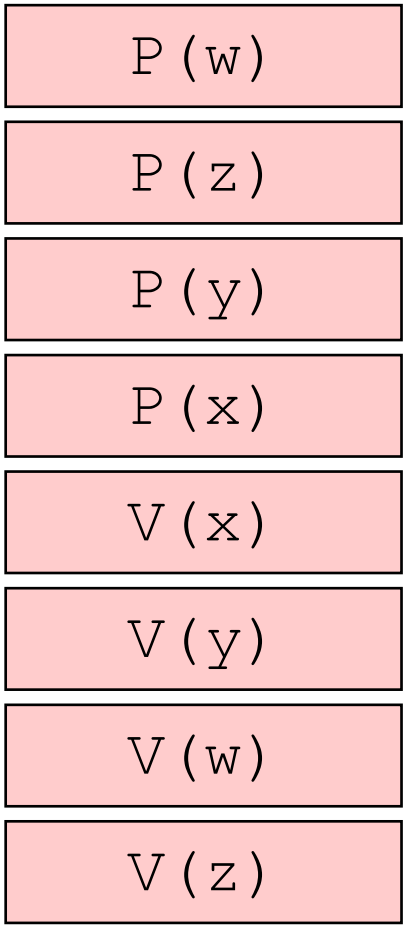
死锁



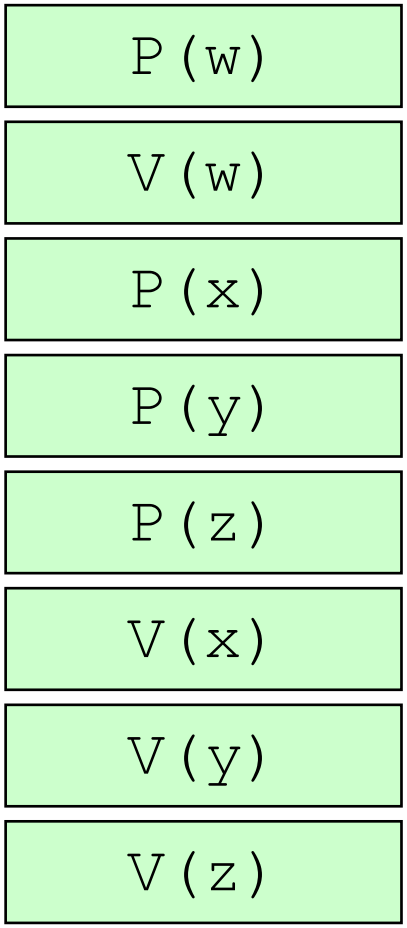
这样会引发死锁吗？

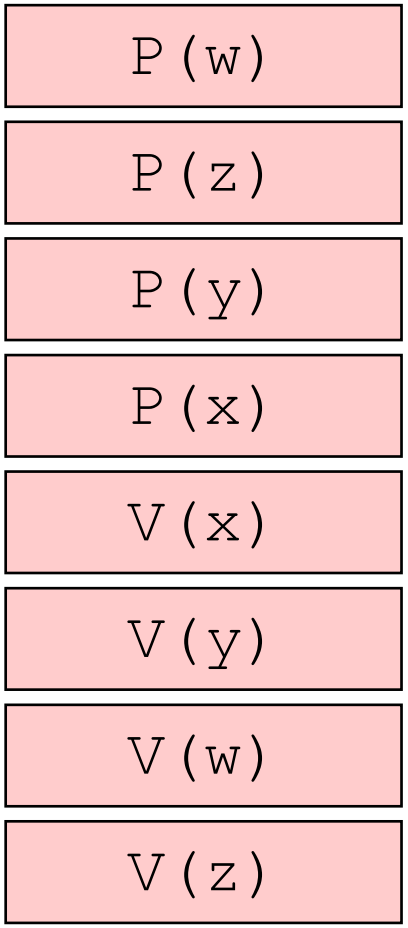


不会

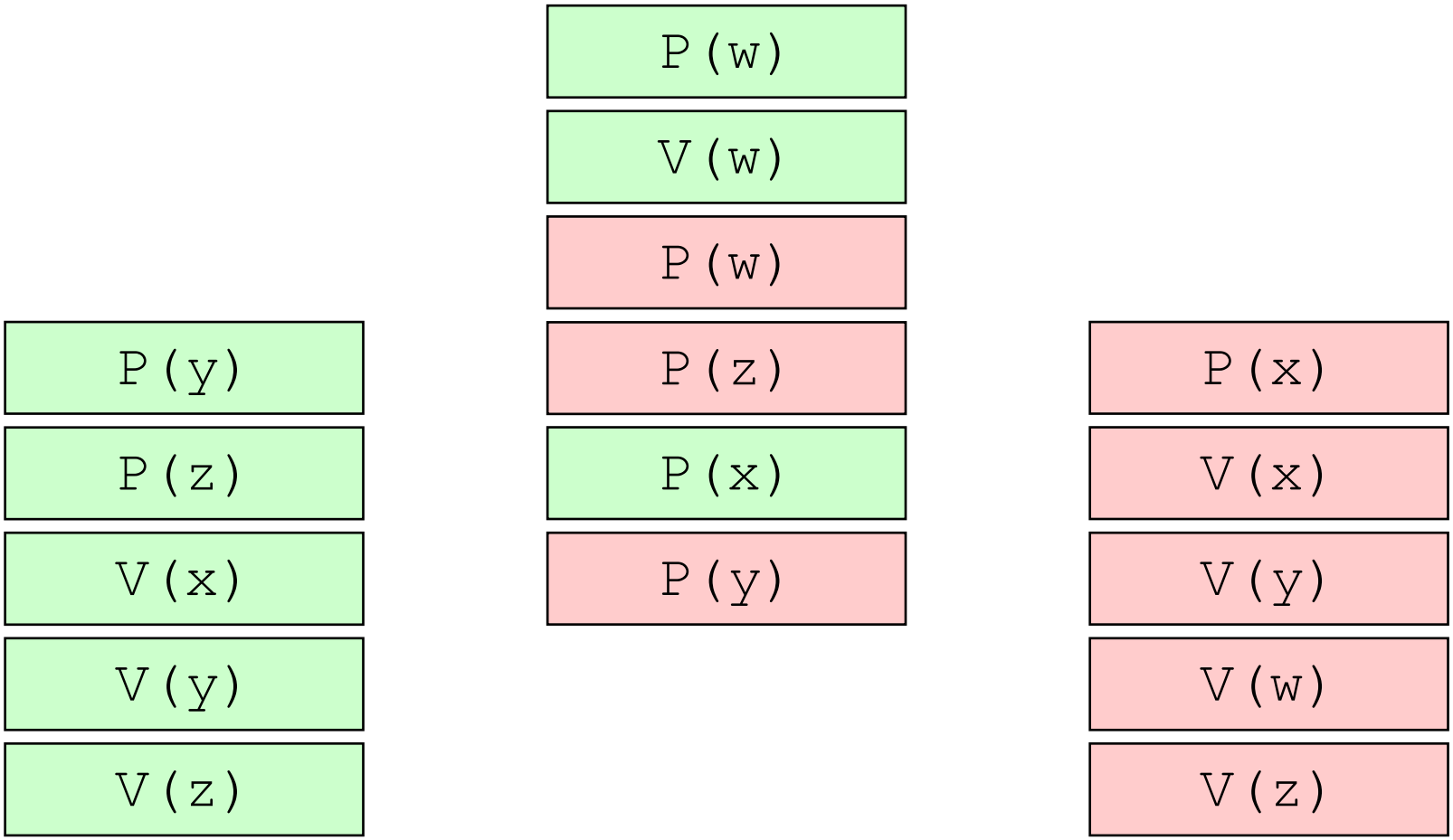


这样会引发死锁吗？





这样会引发死锁吗？



狒狒过峡谷问题

- 一个主修人类学、辅修计算机科学的学生参加了一个研究课题，调查是否可以教会非洲狒狒理解死锁。他找到一处很深的峡谷，在上边固定了一根横跨峡谷的绳索，这样狒狒就可以攀住绳索越过峡谷。同一时刻，只要朝着相同的方向就可以有几只狒狒通过。但如果向东和向西的狒狒同时攀在绳索上那么会产生死锁（狒狒会被卡在中间），由于它们无法在绳索上从另一只的背上翻过去。如果一只狒狒想越过峡谷， 它必须看当前是否有别的狒狒正在逆向通行。利用信号量编写一个避免死锁的程序来解决该问题。不考虑连续东行的狒狒会使得西行的狒狒无限制地等待的情况。

狒狒过峡谷问题

- 读者/写者问题
  - 读者共享
  - 写者互斥
  - 读者写者之间互斥
- 狒狒过峡谷/北大学生过校门闸机
  - 同向共享
  - 异向互斥
  - 可以看作两拨读者

狒狒过峡谷问题

```
void reader()
{
    while(true)
    {
        P(mutex);
        rc = rc + 1;
        if (rc == 1)
            P(w);
        V(mutex);
        /* reading... */
        P(mutex);
        rc = rc - 1;
        if (rc == 0)
            V(w);
        V(mutex);
    }
}
```

```
void writer()
{
    while(true)
    {
        P(w);
        /* writing... */
        V(w);
    }
}
```

w作为0-1互斥信号量来实现读写互斥
第一个读者和写者竞争这个信号量

狒狒过峡谷问题

```
void Westward()
{
    P(W);
    if (CW == 0)
        P(mutex);
    CW = CW + 1;
    V(W);

    cross();

    P(W);
    CW = CW - 1;
    if (CW == 0)
        V(mutex);
    V(W);
}
```

```
void Eastward()
{
    P(E);
    if (CE == 0)
        P(mutex);
    CE = CE + 1;
    V(E);

    cross();

    P(E);
    CE = CE - 1;
    if (CE == 0)
        V(mutex);
    V(E);
}
```

狒狒过峡谷问题

- 读者/写者问题
  - 读者共享
  - 写者互斥
  - 读者写者之间互斥
  - 写者饥饿——第二类读者写者问题
- 狒狒过峡谷/北大学生过校门闸机
  - 同向共享
  - 异向互斥
  - 可以看作两拨读者
  - 可以看作两拨读者——也存在饥饿问题，参考第二类读者写者

考场问题

- 某次考试有30名学生与1名监考老师，该教室的门很狭窄，每次只能通过一人。
- 考试开始前，老师和学生进入考场（有的学生来得比老师早）。
- 当人来齐以后，老师开始发放试卷。
- 拿到试卷后，学生就可以开始答卷。
- 学生可以随时交卷，交卷后就可以离开考场。
- 当所有的学生都上交试卷以后，老师才能离开考场。

全局变量：
stu_count：int类型，表示考场中的学生数量，初值为0
信号量：
mutex_stu_count：保护全局变量，初值为1
mutex_door：保证门每次通过一人，初值为[ ]
mutex_all_present：保证学生都到了，初值为[ ]
mutex_all_handin：保证学生都交了，初值为[ ]
mutex_test[30]：表示学生拿到了试卷，初值均为[ ]

```
Teacher:  // 老师
    (1)
    从门进入考场
    (2)

    (3) // 等待同学来齐
    for (i = 1; i <= 30; i++)
        (4) // 给i号学生发放试卷

    (5) // 等待同学将试卷交齐
    (6)
    从门离开考场
    (7)
```

```
Student(x):  // x号学生
    (8)
    从门进入考场
    (9)

    P(mutex_stu_count);
    stu_count++;
    if (stu_count == 30)
        (10)
    V(mutex_stu_count);

    (11) // 等待拿自己的卷子
    学生答卷

    P(mutex_stu_count);
    stu_count--;
    if (stu_count == 0)
        (12)
    V(mutex_stu_count);

    (13)
    从门离开考场
    (14)
```

全局变量：

stu_count: int类型，表示考场中的学生数量，初值为0

信号量：

mutex_stu_count: 保护全局变量，初值为1

mutex_door: 保证门每次通过一人，初值为[ ]

mutex_all_present: 保证学生都到了，初值为[ ]

mutex_all_handin: 保证学生都交了，初值为[ ]

mutex_test[30]: 表示学生拿到了试卷，初值均为[ ]

```
Teacher:  // 老师
    (1)
    从门进入考场
    (2)

    (3) // 等待同学来齐
    for (i = 1; i <= 30; i++)
        (4) // 给i号学生发放试卷

    (5) // 等待同学将试卷交齐
    (6)
    从门离开考场
    (7)
```

```
Student(x):  // x号学生
    (8)
    从门进入考场
    (9)

    P(mutex_stu_count);
    stu_count++;
    if (stu_count == 30)
        (10)
    V(mutex_stu_count);

    (11) // 等待拿自己的卷子
    学生答卷

    P(mutex_stu_count);
    stu_count--;
    if (stu_count == 0)
        (12)
    V(mutex_stu_count);

    (13)
    从门离开考场
    (14)
```

全局变量：

stu_count：int类型，表示考场中的学生数量，初值为0

信号量：

mutex_stu_count：保护全局变量，初值为1

mutex_door：保证门每次通过一人，初值为[ ]

mutex_all_present：保证学生都到了，初值为[ ]

mutex_all_handin：保证学生都交了，初值为[ ]

mutex_test[30]：表示学生拿到了试卷，初值均为[ ]

```
Teacher:  // 老师
    (1)
    从门进入考场
    (2)

    (3) // 等待同学来齐
    for (i = 1; i <= 30; i++)
        (4) // 给i号学生发放试卷

    (5) // 等待同学将试卷交齐
    (6)
    从门离开考场
    (7)
```

```
Student(x):  // x号学生
    (8)
    从门进入考场
    (9)

    P(mutex_stu_count);
    stu_count++;
    if (stu_count == 30)
        (10)
    V(mutex_stu_count);

    (11) // 等待拿自己的卷子
    学生答卷

    P(mutex_stu_count);
    stu_count--;
    if (stu_count == 0)
        (12)
    V(mutex_stu_count);

    (13)
    从门离开考场
    (14)
```

全局变量：

stu_count：int类型，表示考场中的学生数量，初值为0

信号量：

mutex_stu_count：保护全局变量，初值为1

mutex_door：保证门每次通过一人，初值为1

mutex_all_present：保证学生都到了，初值为[ ]

mutex_all_handin：保证学生都交了，初值为[ ]

mutex_test[30]：表示学生拿到了试卷，初值均为[ ]

```
Teacher:  // 老师
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    (3) // 等待同学来齐
    for (i = 1; i <= 30; i++)
        (4) // 给i号学生发放试卷

    (5) // 等待同学将试卷交齐
    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

```
Student(x):  // x号学生
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_stu_count);
    stu_count++;
    if (stu_count == 30)
        (10)
    V(mutex_stu_count);

    (11) // 等待拿自己的卷子
    学生答卷

    P(mutex_stu_count);
    stu_count--;
    if (stu_count == 0)
        (12)
    V(mutex_stu_count);

    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

全局变量：
stu_count：int类型，表示考场中的学生数量，初值为0
信号量：
mutex_stu_count：保护全局变量，初值为1
mutex_door：保证门每次通过一人，初值为1
mutex_all_present：保证学生都到了，初值为[ ]
mutex_all_handin：保证学生都交了，初值为[ ]
mutex_test[30]：表示学生拿到了试卷，初值均为[ ]

```
Teacher:  // 老师
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    (3) // 等待同学来齐
    for (i = 1; i <= 30; i++)
        (4) // 给i号学生发放试卷

    (5) // 等待同学将试卷交齐
    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

```
Student(x):  // x号学生
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_stu_count);
    stu_count++;
    if (stu_count == 30)
        (10)
    V(mutex_stu_count);

    (11) // 等待拿自己的卷子
    学生答卷

    P(mutex_stu_count);
    stu_count--;
    if (stu_count == 0)
        (12)
    V(mutex_stu_count);

    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

全局变量：

stu_count：int类型，表示考场中的学生数量，初值为0

信号量：

mutex_stu_count：保护全局变量，初值为1

mutex_door：保证门每次通过一人，初值为1

mutex_all_present：保证学生都到了，初值为[ ]

mutex_all_handin：保证学生都交了，初值为[ ]

mutex_test[30]：表示学生拿到了试卷，初值均为0

```
Teacher:  // 老师
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    (3) // 等待同学来齐
    for (i = 1; i <= 30; i++)
        V(mutex_test[i]);

    (5) // 等待同学将试卷交齐
    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

```
Student(x):  // x号学生
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_stu_count);
    stu_count++;
    if (stu_count == 30)
        (10)
    V(mutex_stu_count);

    P(mutex_test[i]);
    学生答卷

    P(mutex_stu_count);
    stu_count--;
    if (stu_count == 0)
        (12)
    V(mutex_stu_count);

    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

全局变量：

stu_count：int类型，表示考场中的学生数量，初值为0

信号量：

mutex_stu_count：保护全局变量，初值为1

mutex_door：保证门每次通过一人，初值为1

mutex_all_present：保证学生都到了，初值为[　　]

mutex_all_handin：保证学生都交了，初值为[　　]

mutex_test[30]：表示学生拿到了试卷，初值均为0

```
Teacher:  // 老师
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    (3) // 等待同学来齐
    for (i = 1; i <= 30; i++)
        V(mutex_test[i]);

    (5) // 等待同学将试卷交齐
    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

```
Student(x):  // x号学生
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_stu_count);
    stu_count++;
    if (stu_count == 30)
        (10)
    V(mutex_stu_count);

    P(mutex_test[i]);
    学生答卷

    P(mutex_stu_count);
    stu_count--;
    if (stu_count == 0)
        (12)
    V(mutex_stu_count);

    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

全局变量：

stu_count：int类型，表示考场中的学生数量，初值为0

信号量：

mutex_stu_count：保护全局变量，初值为1

mutex_door：保证门每次通过一人，初值为1

mutex_all_present：保证学生都到了，初值为**0**

mutex_all_handin：保证学生都交了，初值为[　　]

mutex_test[30]：表示学生拿到了试卷，初值均为0

```
Teacher:  // 老师
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_all_present);
    for (i = 1; i <= 30; i++)
        V(mutex_test[i]);

    (5) // 等待同学将试卷交齐
    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

```
Student(x):  // x号学生
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_stu_count);
    stu_count++;
    if (stu_count == 30)
      V(mutex_all_present);
    V(mutex_stu_count);

    P(mutex_test[i]);
    学生答卷

    P(mutex_stu_count);
    stu_count--;
    if (stu_count == 0)
        (12)
    V(mutex_stu_count);

    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

全局变量：

stu_count：int类型，表示考场中的学生数量，初值为0

信号量：

mutex_stu_count：保护全局变量，初值为1

mutex_door：保证门每次通过一人，初值为1

mutex_all_present：保证学生都到了，初值为0

mutex_all_handin：保证学生都交了，初值为0

mutex_test[30]：表示学生拿到了试卷，初值均为0

```
Teacher:  // 老师
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_all_present);
    for (i = 1; i <= 30; i++)
        V(mutex_test[i]);

    P(mutex_all_handin);
    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

```
Student(x):  // x号学生
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_stu_count);
    stu_count++;
    if (stu_count == 30)
      V(mutex_all_present);
    V(mutex_stu_count);

    P(mutex_test[i]);
    学生答卷

    P(mutex_stu_count);
    stu_count--;
    if (stu_count == 0)
      V(mutex_all_handin);
    V(mutex_stu_count);

    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

考场问题

- 问：改成右边这样，运行结果正确吗？
- 运行结果正确，但没有保护好全局变量

```
Teacher:  // 老师
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_all_present);
    for (i = 1; i <= 30; i++)
        V(mutex_test[i]);

    P(mutex_all_handin);
    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

```
Student(x):  // x号学生
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_stu_count);
    stu_count++;
    V(mutex_stu_count);
    if (stu_count == 30)
      V(mutex_all_present);

    P(mutex_test[i]);
    学生答卷

    P(mutex_stu_count);
    stu_count--;
    V(mutex_stu_count);
    if (stu_count == 0)
      V(mutex_all_handin);

    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

考场问题

- 问：能否将两个信号量合为一个？
- 初始值为0

```
Teacher:  // 老师
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_all);
    for (i = 1; i <= 30; i++)
        V(mutex_test[i]);

    P(mutex_all);
    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

```
Student(x):  // x号学生
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_stu_count);
    stu_count++;
    if (stu_count == 30)
        V(mutex_all);
    V(mutex_stu_count);

    P(mutex_test[i]);
    学生答卷

    P(mutex_stu_count);
    stu_count--;
    if (stu_count == 0)
        V(mutex_all);
    V(mutex_stu_count);

    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

考场问题

- 问：将两个信号量合为一个以后，能否像刚才一样缩小保护全局变量的范围？
- 运行结果不正确

```
Teacher:  // 老师
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_all);
    for (i = 1; i <= 30; i++)
        V(mutex_test[i]);

    P(mutex_all);
    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

```
Student(x):  // x号学生
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_stu_count);
    stu_count++;
    V(mutex_stu_count);
    if (stu_count == 30)
        V(mutex_all);

    P(mutex_test[i]);
    学生答卷

    P(mutex_stu_count);
    stu_count--;
    V(mutex_stu_count);
    if (stu_count == 0)
        V(mutex_all);

    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

考场问题

- 问：有没有不用全局变量的办法？

```
Teacher:  // 老师
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_all_present);
    for (i = 1; i <= 30; i++)
        V(mutex_test[i]);

    P(mutex_all_handin);
    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

```
Student(x):  // x号学生
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_stu_count);
    stu_count++;
    if (stu_count == 30)
      V(mutex_all_present);
    V(mutex_stu_count);

    P(mutex_test[i]);
    学生答卷

    P(mutex_stu_count);
    stu_count--;
    if (stu_count == 0)
      V(mutex_all_handin);
    V(mutex_stu_count);

    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

考场问题

- 问：有没有不用全局变量的办法？
- 初值均为0

```
Teacher:  // 老师
    P(mutex_door);
    从门进入考场
    V(mutex_door);
    for (i = 1; i <= 30; i++)
        P(mutex_arrive[i]);
    for (i = 1; i <= 30; i++)
        V(mutex_test[i]);
    for (i = 1; i <= 30; i++)
        P(mutex_leave[i]);
    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

```
Student(x):  // x号学生
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    V(mutex_arrive[i]);

    P(mutex_test[i]);
    学生答卷

    V(mutex_leave[i]);
    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

考场问题

- 问：有没有不用信号量数组的办法？

```
Teacher:  // 老师
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_all_present);
    for (i = 1; i <= 30; i++)
        V(mutex_test[i]);

    P(mutex_all_handin);
    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

```
Student(x):  // x号学生
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_stu_count);
    stu_count++;
    if (stu_count == 30)
      V(mutex_all_present);
    V(mutex_stu_count);

    P(mutex_test[i]);
    学生答卷

    P(mutex_stu_count);
    stu_count--;
    if (stu_count == 0)
      V(mutex_all_handin);
    V(mutex_stu_count);

    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

考场问题

- 问：有没有不用信号量数组的办法？
- 初值为0

```
Teacher:  // 老师
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_all_present);
    for (i = 1; i <= 30; i++)
        V(mutex_test);

    P(mutex_all_handin);
    P(mutex_door);
    从门离开考场
    V(mutex_door);
```

```
Student(x):  // x号学生
    P(mutex_door);
    从门进入考场
    V(mutex_door);

    P(mutex_stu_count);
    stu_count++;
    if (stu_count == 30)
      V(mutex_all_present);
    V(mutex_stu_count);

    P(mutex_test);
    学生答卷

    P(mutex_stu_count);
    stu_count--;
    if (stu_count == 0)
      V(mutex_all_handin);
    V(mutex_stu_count);

    P(mutex_door);
    从门离开考场
    V(mutex_door);
```