并发编程

金超 章梓立

2021-12-21

孙英博

2023-12

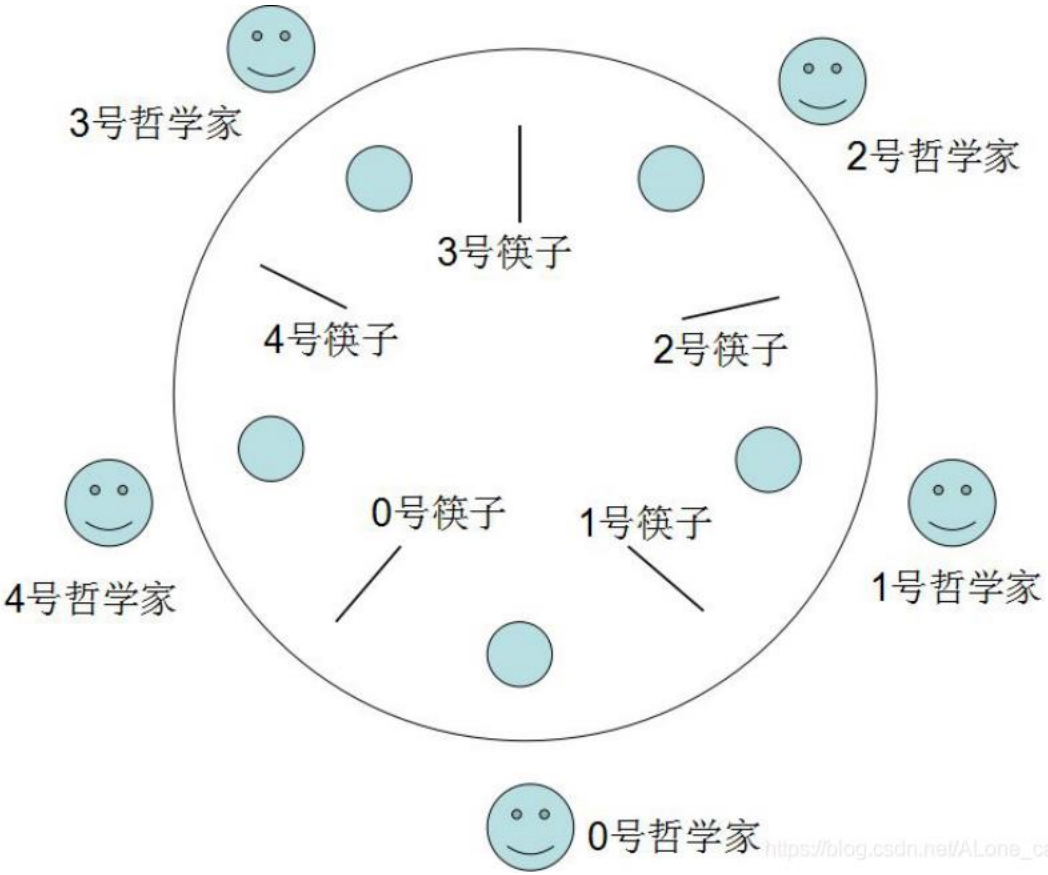
其他并发问题

- 竞争：定义详见讲义
- 死锁：一个程序等待一个不可能为真的条件
- 线程不安全：在多个线程调用这个函数的时候，不会产生错误的结果
- 饥饿：见读者-写者问题

死锁产生的必要条件

- 互斥使用:一个资源每次只能给一个进程使用
- 占有且等待:进程在申请新的资源的同时，保持对原来资源的占有
- 不可抢占
- 循环等待，存在一个等待队列 Pi 等 Pi+1
  (占有Pi资源，请求Pi+1资源)

哲学家就餐问题



如何解决死锁？

- 互斥使用:一个资源每次只能给一个进程使用
- 占有且等待:进程在申请新的资源的同时，保持对原来资源的占有
- 不可抢占
- 循环等待，存在一个等待队列 Pi 等 Pi+1

- 两个哲学家一起使用叉子，把独占资源变为共享资源 (x
- 每次同时申请两个叉子(破坏“占有且申请”条件)。
- 给哲学家安排优先级，优先级高的可以抢占低的叉子 (破坏 不可抢占)
- 给叉子编号，每个哲学家需要先申请编号小的叉子 (破坏“循环等待”条件);

如何避免死锁

- 有一个统计：死锁在所有并发问题中占40%左右
- 书P722页：

互斥锁加锁顺序规则：给定所有互斥操作的一个全序，如果每个线程都是以一种顺序获得互斥锁并以相反的顺序释放，那么这个程序就是无死锁的。

例如，我们可以通过这样的方法来解决图 12-44 中的死锁问题：在每个线程中先对 s 加锁，然后再对 t 加锁。图 12-45 展示了得到的进度图。

- **释放锁的顺序并不重要**，只要可以保证每个资源申请者都以一个同样的顺序获得锁，就可以保证没有死锁
- E.g.:
- 线程一: P(A) P(B) P(C) V(B) V(A) V(C)
- 线程二: P(A) P(B) P(C) V(A) V(B) V(C)

如何避免死锁

- 用进程图理解解锁顺序并不重要

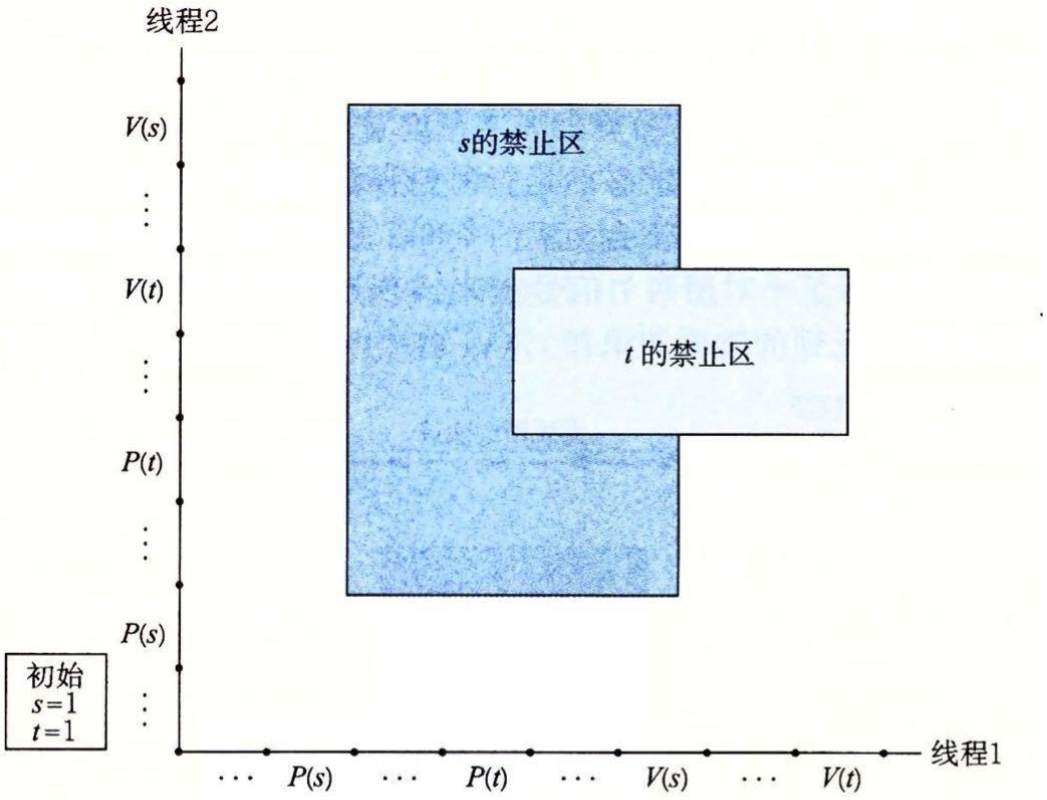


图 12-45　一个无死锁程序的进度图

生产-消费者问题

- 例子：共享buffer

Practice Problem 12.9 (solution page 1074)

Let p denote the number of producers, c the number of consumers, and n the buffer size in units of items. For each of the following scenarios, indicate whether the mutex semaphore in `sbuf_insert` and `sbuf_remove` is necessary or not.

A. $p = 1, c = 1, n > 1$

B. $p = 1, c = 1, n = 1$

C. $p > 1, c > 1, n = 1$

code/conc/sbuf.c

```
#include "csapp.h"
#include "sbuf.h"

/* Create an empty, bounded, shared FIFO buffer with n slots */
void sbuf_init(sbuf_t *sp, int n)
{
    sp->buf = Calloc(n, sizeof(int));
    sp->n = n;                      /* Buffer holds max of n items */
    sp->front = sp->rear = 0;       /* Empty buffer iff front == rear */
    Sem_init(&sp->mutex, 0, 1);     /* Binary semaphore for locking */
    Sem_init(&sp->slots, 0, n);     /* Initially, buf has n empty slots */
    Sem_init(&sp->items, 0, 0);     /* Initially, buf has zero data items */
}

/* Clean up buffer sp */
void sbuf_deinit(sbuf_t *sp)
{
    Free(sp->buf);
}

/* Insert item onto the rear of shared buffer sp */
void sbuf_insert(sbuf_t *sp, int item)
{
    P(&sp->slots);                          /* Wait for available slot */
    P(&sp->mutex);                          /* Lock the buffer */
    sp->buf[(++sp->rear)%(sp->n)] = item;   /* Insert the item */
    V(&sp->mutex);                          /* Unlock the buffer */
    V(&sp->items);                          /* Announce available item */
}

/* Remove and return the first item from buffer sp */
int sbuf_remove(sbuf_t *sp)
{
    int item;
    P(&sp->items);                          /* Wait for available item */
    P(&sp->mutex);                          /* Lock the buffer */
    item = sp->buf[(++sp->front)%(sp->n)];  /* Remove the item */
    V(&sp->mutex);                          /* Unlock the buffer */
    V(&sp->slots);                          /* Announce available slot */
    return item;
}
```

code/conc/sbuf.c

Figure 12.25 SBUF: **A package for synchronizing concurrent access to bounded buffers.**

读者-写者问题

- 例子：最简单的例子：badcnt.c

```c
/* Global shared variable */
volatile long cnt = 0; /* Counter */

int main(int argc, char **argv)
{
    long niters;
    pthread_t tid1, tid2;

    niters = atoi(argv[1]);
    Pthread_create(&tid1, NULL,
        thread, &niters);
    Pthread_create(&tid2, NULL,
        thread, &niters);
    Pthread_join(tid1, NULL);
    Pthread_join(tid2, NULL);
```

```c
/* Thread routine */
void *thread(void *vargp)
{
    long i, niters =
                *((long *)vargp);

    for (i = 0; i < niters; i++)
        cnt++;

    return NULL;
}
```

- 全是写者，没有读者

读者-写者问题

- 当然，也可以很容易地给出一个全是读者没有写者的例子

```c
int read() {
    return count;
}

void reader(void* argp) {
    int cnt = read_cnt();
    printf("%d\n", cnt);
}

int main() {
    pthread_t tid;
    for (int i = 0; i < 10; i++) {
        pthread_create(&tid, NULL, reader, NULL);
    }
    return 0;
}
```

- 以上是两种最简单的情况

读者-写者问题

- 一般地，需要满足：
  - 写者之间互斥
  - 写者和读者互斥
  - 多个读者可以同时进行读操作
- 具体实现

读者-写者问题

- 第一类读者写者问题
- 问题：
- 写者饥饿

```
/* Global variables */
int readcnt;    /* Initially = 0 */
sem_t mutex, w; /* Both initially = 1 */

void reader(void)
{
    while (1) {
        P(&mutex);
        readcnt++;
        if (readcnt == 1) /* First in */
            P(&w);
        V(&mutex);

        /* Critical section */
        /* Reading happens  */

        P(&mutex);
        readcnt--;
        if (readcnt == 0) /* Last out */
            V(&w);
        V(&mutex);
    }
}

void writer(void)
{
    while (1) {
        P(&w);

        /* Critical section */
        /* Writing happens  */

        V(&w);
    }
}
```

Figure 12.26 **Solution to the first readers-writers problem.** Favors readers over writers.

读者-写者问题

- 第二类解法
- 写者优先

```
int readcnt, writecnt;       // Initially 0
sem_t rmutex, wmutex, r, w; // Initially 1
void reader(void)
{
  while (1) {
    P(&r);
    P(&rmutex);
    readcnt++;
    if (readcnt == 1) /* First in */
      P(&w);
    V(&rmutex);
    V(&r)

    /* Reading happens here */

    P(&rmutex);
    readcnt--;
    if (readcnt == 0) /* Last out */
      V(&w);
    V(&rmutex);
  }
}
```


Second Readers-Writers

```
void writer(void)
{
  while (1) {
    P(&wmutex);
    writecnt++;
    if (writecnt == 1)
        P(&r);
    V(&wmutex);

    P(&w);
    /* Writing here */
    V(&w);

    P(&wmutex);
    writecnt--;
    if (writecnt == 0);
        V(&r);
    V(&wmutex);
  }
}
```

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);
```