1. 读者写者问题(写者优先): 1)共享读; 2)互斥写、读写互斥; 3)写者优先于读者(一旦有写者，则后续读者必须等待，唤醒时优先考虑写者)。

```
int wcount = 0;       //用于记录写者数量
int rcount = 0;       //用于记录读者数量
semaphore rmutex = 1; //用于读者进程互斥修改rcount
semaphore wmutex = 1; //用于写者进程互斥修改wcount
semaphore file = 1;   //用于读者写者互斥访问file
semaphore read = 1;   //用于阻塞读者进程，实现写者优先

cobegin
writer() {
    while (true) {
        P(wmutex);
        if (wcount == 0)
            P(read);
        wcount++;
        V(wmutex);

        P(file); //写者互斥访问文件
        writing();
        V(file);

        P(wmutex);
        wcount--;
        if (wcount == 0)
            V(read);
        V(wmutex);
    }
}

reader() {
    while (true) {
        P(read); //检查写者队列是否为空。
        P(rmutex);
        if (rcount == 0)
            P(file); //申请文件资源
        rcount++;
        V(rmutex);
        V(read);

        reading();

        P(rmutex);
        rcount--;
        if (rcount == 0)
            V(file);
        V(rmutex);
    }
}
coend
```

2. 寿司店问题。

```
semaphore mutex = 1;    // 保证客人到达与离开时计算的互斥
semaphore block = 0;    // 用于等待队列
bool must_wait = false; // 为真表示寿司店已满需等待
int eating = 0;         // 记录在寿司店就餐的线程数
int waiting = 0;        // 记录在寿司店等待的线程数
cobegin
comein() {
    while (true) {
        P(mutex);
        if (must_wait) {
            waiting++;
            V(mutex);
            P(block);
        } else {
            eating++;
            if (eating == 5)
                must_wait = true;
            else
                must_wait = false;
            V(mutex);
        }

        eat();

        P(mutex);
        eating--;
        if (eating == 0) {
            int n = min(5, waiting);
            waiting -= n;
            eating += n;
            if (eating == 5)
                must_wait = true;
            else
                must_wait = false;
            while (n--)
                V(block);
        }
        V(mutex);
    }
}
coend
```

3. 三个进程 P1、P2、P3 互斥使用一个包含 N(N > 0)个单元的缓冲区。

```
semaphore mutex = 1;          // 缓冲区操作互斥信号量
semaphore empty = N;          // 缓冲区空单元数量信号量
semaphore odd = 0, even = 0; // 奇偶数信号量

cobegin
P1() {
    while (true) {
        x = produce();
        P(empty);
        P(mutex);
        put();
        V(mutex);
        if (x % 2 == 0)
            V(even);
        else
            V(odd);
    }
}

P2() {
    while (true) {
        P(odd);
        P(mutex);
        getodd();
        V(mutex);
        V(empty);
        countodd();
    }
}

P3() {
    while (true) {
        P(even);
        P(mutex);
        geteven();
        V(mutex);
        V(empty);
        counteven();
    }
}
coend
```

4. 搜索-插入-删除问题。

```
int scount = 0;
int icount = 0;
int dcount = 0;
semaphore smutex = 1; // 用于搜索线程修改 scount
semaphore imutex = 1; // 用于插入线程修改 icount

semaphore sd = 1; // 用于搜索和删除线程互斥
semaphore id = 1; // 用于插入和删除线程互斥

semaphore insert = 1; // 用于插入线程之间互斥
semaphore delete = 1; // 用于删除线程之间互斥

cobegin
searcher() {
    while (true) {
        P(smutex);
        if (scount == 0) {
            P(sd);
        }
        scount++;
        V(smutex);

        searching();

        P(smutex);
        scount--;
        if (scount == 0) {
            V(sd);
        }
        V(smutex);
    }
}
```

```

inserter() {
    while (true) {
        P(imutex);
        if (icount == 0) {
            P(id);
        }
        icount++;
        V(imutex);

        P(insert);
        inserting();
        V(insert);

        P(imutex);
        icount--;
        if (icount == 0) {
            V(id);
        }
        V(imutex);
    }
}

deleter() {
    while (true) {
        P(sd);
        P(id);
        P(delete);
        deleting();
        V(delete);
        V(id);
        V(sd);
    }
}
coend
```