

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)。

```

1  int wcount = 0;           //用于记录写者数量
2  int rcount = 0;           //用于记录读者数量
3  semaphore rmutex = 1;    //用于读者进程互斥修改rcount
4  semaphore wmutex = 1;    //用于写者进程互斥修改wcount
5  semaphore file = 1;      //用于读者写者互斥访问file
6  semaphore read = 1;      //用于阻塞读者进程, 实现写者优先
7
8  cobegin
9  writer() {
10     while (true) {
11         P(wmutex);
12         if (wcount == 0)
13             P(read);
14         wcount++;
15         V(wmutex);
16
17         P(file); //写者互斥访问文件
18         writing();
19         V(file);
20
21         P(wmutex);
22         wcount--;
23         if (wcount == 0)
24             V(read);
25         V(wmutex);
26     }
27 }
28
29 reader() {
30     while (true) {
31         P(read); //检查写者队列是否为空。
32         P(rmutex);
33         if (rcount == 0)
34             P(file); //申请文件资源
35         rcount++;
36         V(rmutex);
37         V(read);
38
39         reading();

```

```

40
41     P(rmutex);
42     rcount--;
43     if (rcount == 0)
44         V(file);
45     V(rmutex);
46 }
47 }
48 coend

```

2. 寿司店问题。

```

1  semaphore mutex = 1;    // 保证客人到达与离开时计算的互斥
2  semaphore block = 0;    // 用于等待队列
3  bool must_wait = false; // 为真表示寿司店已满需等待
4  int eating = 0;         // 记录在寿司店就餐的线程数
5  int waiting = 0;        // 记录在寿司店等待的线程数
6  cobegin
7  comein() {
8      while (true) {
9          P(mutex);
10         if (must_wait) {
11             waiting++;
12             V(mutex);
13             P(block);
14         } else {
15             eating++;
16             if (eating == 5)
17                 must_wait = true;
18             else
19                 must_wait = false;
20             V(mutex);
21         }
22
23         eat();
24
25         P(mutex);
26         eating--;
27         if (eating == 0) {
28             int n = min(5, waiting);
29             waiting -= n;
30             eating += n;
31             if (eating == 5)
32                 must_wait = true;

```

```

33         else
34             must_wait = false;
35         while (n--)
36             V(block);
37     }
38     V(mutex);
39 }
40 }
41 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。

```

1  semaphore mutex = 1;           // 缓冲区操作互斥信号量
2  semaphore empty = N;           // 缓冲区空单元数量信号量
3  semaphore odd = 0, even = 0;   // 奇偶数信号量
4
5  cobegin
6  P1() {
7      while (true) {
8          x = produce();
9          P(empty);
10         P(mutex);
11         put();
12         V(mutex);
13         if (x % 2 == 0)
14             V(even);
15         else
16             V(odd);
17     }
18 }
19
20 P2() {
21     while (true) {
22         P(odd);
23         P(mutex);
24         getodd();
25         V(mutex);
26         V(empty);
27         countodd();
28     }
29 }
30
31 P3() {
32     while (true) {

```

```

33     P(even);
34     P(mutex);
35     geteven();
36     V(mutex);
37     V(empty);
38     counteven();
39 }
40 }
41 coend

```

4. 搜索-插入-删除问题。

```

1  int scout = 0;
2  int icount = 0;
3  int dcount = 0;
4  semaphore smutex = 1; // 用于搜索线程修改 scout
5  semaphore imutex = 1; // 用于插入线程修改 icount
6
7  semaphore sd = 1; // 用于搜索和删除线程互斥
8  semaphore id = 1; // 用于插入和删除线程互斥
9
10 semaphore insert = 1; // 用于插入线程之间互斥
11 semaphore delete = 1; // 用于删除线程之间互斥
12
13 cobegin
14 searcher() {
15     while (true) {
16         P(smutex);
17         if (scout == 0) {
18             P(sd);
19         }
20         scout++;
21         V(smutex);
22
23         searching();
24
25         P(smutex);
26         scout--;
27         if (scout == 0) {
28             V(sd);
29         }
30         V(smutex);
31     }
32 }

```

```

33
34 inserter() {
35     while (true) {
36         P(imutex);
37         if (icount == 0) {
38             P(id);
39         }
40         icount++;
41         V(imutex);
42
43         P(insert);
44         inserting();
45         V(insert);
46
47         P(imutex);
48         icount--;
49         if (icount == 0) {
50             V(id);
51         }
52         V(imutex);
53     }
54 }
55
56 deleter() {
57     while (true) {
58         P(sd);
59         P(id);
60         P(delete);
61         deleting();
62         V(delete);
63         V(id);
64         V(sd);
65     }
66 }
67 coend

```