作业四

> 22373407　王飞阳

1. 读者写者问题

```
Semaphore write_write = 1;       // 保证写-写互斥
Semaphore read_write = 1;        // 保证读-写互斥
Semaphore writers_mutex = 1;     // 保证访问共享变量writers互斥
Semaphore readers_mutex = 1;     // 保证访问共享变量readers互斥
Semaphore write_pendings = 1;    // 保证写者优先于读者
int writers = 0, readers = 0;    // 计数写者和读者的数量

// 读者
Readers:
while(true) {
    P(write_pending);
    P(read_write);
    P(readers_mutex);
        readers++;
        if (readers == 1)
            P(write_write);
    V(readers_mutex);
    V(read_write);
    V(write_pending);
    read();
    P(readers_mutex);
        readers--;
        if (readers == 0)
            V(write_write);
    V(readers_mutex);
}

//写者
Writers:
while(true) {
    P(writers_mutex);
        writers++;
        if (writers == 1)
            P(read_write);
    V(writers_mutex);
    P(write_write);
        write();
    V(write_write);
    P(writers_mutex);
        writers--;
        if (writers == 0)
            V(read_write);
    V(writers_mutex);
}
```

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

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);
    }
    sit_and_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);
}
```

3. 进门问题

```
Semaphore mutex = 1;         // 信号量mutex代表进门互斥，初始化为1，保证互斥访问。
Semaphore block = 0;         // 信号量barrier作为屏障，初始化为0，使得员工必须等待。
int count = 0;                // 计数器，表示当前已经刷卡的员工数目。

While(true) {
  P(mutex)
  count += 1
  if count == 5 {
    V(barrier)
  }
  V(mutex)
  P(barrier)
  P(mutex)
  enter_office();
  V(mutex)
```

```
  if (count == 5) {
    close_the_door();
     break;
  }
}
```

4. 搜索-插入-删除问题

```
Semaphore search_search = 1;     // 确保搜索线程之间互斥访问
Semaphore search_delate = 1;     // 确保搜索线程与删除线程之间互斥
Semaphore insert_insert = 1;     // 确保插入线程之间互斥
Semaphore insert_delate = 1;     // 确保插入线程与删除线程之间互斥
int searchers = 0;               // 记录搜索线程数量
int inserters = 0;               // 记录插入线程数量

Seachers:
while(true) {
    P(search_search);
        searchers++;
        if (searchers == 1)
            P(search_delate);
    V(search_search);
    search();
    P(search_search);
        searchers--;
        if (searcher == 0)
            V(search_delate);
    V(search_search);
}

Inserters:
while(true) {
    P(insert_insert);
        inserters++;
        if (inserters == 1)
            P(insert_delate);
    V(insert_insert);
    P(insert_insert);
        insert();
    V(insert_insert);
    P(insert_insert);
        if (inserters == 0)
            V(insert_delate);
    V(insert_insert);
}

Deleters:
while(true) {
    P(search_delate);
    P(insert_delate);
        delate();
    V(insert_delate);
    V(search_delate);
```

```
}
```