# OS第四次理论作业

## 1. 读写者问题

```
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

Readers:
while(true) {
    P(write_pending);  //限制读者 当有写者时 其他读者将无法继续往下执行
    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);
}

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(readBlock);
    V(writers_mutex);
}
```

## 2.寿司店问题

```
Semaphore access = 5;
Semaphore seat = 5;
Semaphore mutex = 1;
bool blk = false; int cnt = 5, rdy = 0;

void Consumer(){
    P(mutex);
    blk = (--cnt < 0);
    V(mutex);
    P(seat);
    P(access);
    set_eat();
    V(seat);
    P(mutex);
    if (++rdy >= 5 || !blk)
        while (rdy--) V(access), ++cnt;
    blk = (cnt > 0);
    V(mutex);
}
```

## 3.缓冲区问题

```
Semaphore mutex = 1;
Semaphore odd = 0;
Semaphore even = 1;
Semaphore empty = N;
P1:
while(true) {
    P(empty);
    integer = produce();
    P(mutex);
    put();
    V(mutex);
    if (integer % 2 == 0)
        V(even);
    else
        V(odd);
}

P2:
while(true) {
    P(odd);
    P(mutex);
    gedodd();
    V(mutex);
    V(empty);
```

```
    countodd();
}

P3:
while(true) {
    P(even);
    P(mutex);
    gedeven();
    V(mutex);
    V(empty);
    counteven();
}
```

## 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,  inserters = 0;  // 记录搜索线程数量，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);
}
```