读者写者算法-写者优先

```
// Reader
noReaders.wait()
    readSwitch.lock(noWriters)
noReaders.signal()
    ///////////////////////////////
    // Critical Section for Readers
    ///////////////////////////////
    readSwitch.unlock(noWriters)

// Writer
writeSwitch.lock(noReaders)
    noWriters.wait()
        ///////////////////////////////
        // Critical Section for Writers
        ///////////////////////////////
    noWriters.signal()
writeSwitch.unlock(noReaders)

// Config.
readSwitch = Lightswitch()
writeSwitch = Lightswitch()
noReaders = Semaphore(1)
noWriters = Semaphore(1)
```

> 最初的信号量都是解锁态。若Reader在临界区，会给noWriter上锁，但是不会给noReader上锁。如果这时候Writer到来，则会给noReader加锁，会让后续读者排队在noReader。当最后一个读者离开，他会signal noWriter，这时写者可以进入。
>
> 当写者进入临界区，则同时拥有noReader和noWriter两个锁。一方面，其他读者和写者不能同时访问临界区；另一方面，writeSwitch允许其他写者通过，并在noWriter等待，但是读者只能在noReader等待。以此所有的排队写者都可以通过临界区，而不需要signal noReader。当最后一个写者离开，noReader才解锁，写者才能进入。

寿司店问题

```
eating = 0
waiting = 0
mutex = Semaphore(1)
block = Semaphore(0)
```

```
must_wait = False

mutex.wait()
if must_wait:
    waiting += 1
    mutex.signal()
    block.wait()
else:
    eating += 1
    must_wait = (eating == 5)
    mutex.signal()

mutex.wait()
eating -= 1
if eating == 0:
    n = min(5, waiting)
    waiting -= n
    eating += n
    must_wait = (eating == 5)
    block.signal(n)
mutex.signal()
```

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;
semaphore   odd = 0;
semaphore   even = 1;
semaphore   empty = N;

P1() {
    while(1) {
        p(empty);
        num = produce();
        p(mutex);
        put();
        v(mutex);
        if(num % 2 == 0)
            v(even);
        else
            v(odd);
    }
}

```

```
P2() {
    while(1) {
        p(odd);
        p(mutex);
        getodd();
        countodd();
        v(mutex);
        v(empty);
    }
}

P3() {
    while(1) {
        p(even);
        p(mutex);
        geteven();
        counteven();
        v(mutex);
        v(empty);
    }
}
```

搜索-插入-删除问题

```
# LightSwitch
insertMutex = Semaphore(1)
noSearcher = Semaphore(1)
noInserter = Semaphore(1)
searchSwitch = Lightswitch()
insertSwitch = Lightswitch()

# Searcher
searchSwitch.wait(noSearcher)
##################
# Critical Section
##################
searchSwitch.signal(noSearcher)

# Inserter
insertSwitch.wait(noInserter)
insertMutex.wait()
##################
# Critical Section
##################
insertMutex.signal()
insertMutex.signal(noInserter)

# Deleter
noSearcher.wait()
noInserter.wait()
##################
# Critical Section
##################
noInserter.signal()
noSearcher.signal()

# Config.
insertMutex = Semaphore(1)
noSearcher = Semaphore(1)
noInserter = Semaphore(1)

searcher = 0
searcherMutex = Semaphore(1)
inserter = 0
inserterMutex = Semaphore(1)

# Searcher
searcherMutex.wait()
searcher += 1
if searcher == 1:
    noSearcher.wait()
searcherMutex.signal()

# Deleter
noSearcher.wait()
noInserter.wait()
##################
# Critical Section
##################
noInserter.signal()
noSearcher.signal()

##################
# Critical Section
##################

searcherMutex.wait()
searcher -= 1
if searcher == 0:
    noSearcher.signal()
searcherMutex.signal()

# Inserter
inserterMutex.wait()
```

```
inserter += 1
if inserter == 1:
    noInserter.wait()
inserterMutex.signal()

inserterMutex.wait()
##################
# Critical Section
##################
inserterMutex.signal()

inserterMutex.wait()
inserter -= 1
if inserter == 0:
    noInserter.signal()
inserterMutex.signal()
```