

协同读者/写者问题的写者优先方案

- 至少有一个写进程时, 禁止所有的读进程, 但不禁止其它写进程;
- 定义变量 writeCount 记录写进程的个数;
- 对 writeCount 的访问需互斥进行;
- 读进程和写进程不能全部排在一个等待队列中。

mutex 表示对 writeCount 互斥访问;

wmutex 表示写进程对缓冲区的互斥访问

```
semaphore mutex = 1, wmutex = 1, readBlock = 1;
```

```
int writeCount = 0;
```

```
void Writer()
```

```
{
```

```
    While(true){
```

```
        P(mutex);
```

```
        writeCount++;
```

```
        if(writeCount == 1)
```

```
            P(readBlock);
```

```
        V(mutex);
```

```
        P(wmutex);
```

```
        Access(resource);
```

```
        V(wmutex);
```

```
        P(mutex);
```

```
        writeCount--;
```

```
        if(writeCount == 0)
```

```
        V(readBlock);

        V(wmutex);

    }

}
```

```
void Reader()
{
    while(true){

        P(readBlock);

        Access(resource);

        V(readBlock);

    }

}
```

1. 当一个读者在读时, 第一个写进程在 readBlock 上等待, 而第二个写进程却可以进入临界去写, 因为 wmutex 初值为 1;
2. 以上算法并不能让多个读进程同时读。readBlock 的初值为 1, 当一个读进程在读的时候, 其它的读进程在 readBlock 信号量上等待;