# **README**

## 说明

### 添加系统调用

- sys\_print\_str(char\* str)
   封装打印字符串操作。但考虑到A-F打印的颜色不同,在此方法内加入根据进程pid打印不同颜色的功能
- sys\_sleep\_milli(int milli\_seconds)

```
p_proc_ready->sleep_ticks=milli_seconds / (1000 / HZ);
schedule();
```

- sys\_p
- sys\_v

### 实现读者优先

在main.c中

- init\_semRF()初始化相关变量及信号量
- 读者进程基本流程RFreader()

```
      P(读计数器);//读计数器: 第一个读时获取锁,随后一个读完时释放锁,实现同时读取读计数器++;

      if (rCount == 0){

      P(&resourceSem);//资源锁

      }

      V(读计数器);

      // 读文件

      P(&readersSem);//文件同时能有读者数的信号量读文件();

      V(&readersSem);

      P(读计数器);

      if (rCount == 0){

      V(&resourceSem);//释放资源锁

      }

      V(读计数器);
```

• 写者进程基本流程RFwriter()

```
获取资源锁;
读文件;
释放资源锁;
```

• 问题:任何一个读者来了应该总是会插队:

当此时有100个的写请求在排队进行写操作,这个时候一个读请求进入,按照这个答案的写法,这个读请求并不能直接插队,必须等待前面的100个写者完成操作才能开始读操作;然而如果此时进行的是读操作,这时候到来的所有的读请求才可以插队。

外面多加一个信号量,让写者先阻塞在这个信号量上,这样一次只有一个写者能进入临界区准备写。当读者想要读的时候前面总是至多只有一个写者,其他写者都卡在外面的信号量上。

○ 解决: 写者进程

```
P(&writerQueueSem);
基本流程;
V(&writerQueueSem);
```

#### • 解决写者饥饿:

当有读进程正在读共享文件时,有写进程请求访问,这时将会禁止后进读进程的请求,等到已在共享文件的读进程执行完毕,立即让写进程进入执行,只有在无写进程执行的情况下才允许读进程的再次运行,即解决了写进程饥饿的问题。

使用信号量 wHungSem 实现。具体见代码.

### 实现写者优先

写者应该尽早开始写操作。

在上面的程序中,通过 readTry 来实现写者优先,resource 是资源锁,readTry 相当于大门锁。你只有进得了大门,才能拿得了资源,所以我觉得 readTry 改名为 door 更贴切。读者计数结束之后,就把门锁放在一边,这个时候如果写者来了,那么写者就会把门锁拾起来,等读者读操作结束释放资源后,写者就可以开始写了。同样地,由于是最后一个写者释放门锁,所以,只有当所有的写者完成操作后,读者才能进入,所以读者可能饿死。门锁是关键啊!

同理,为避免一大群读者在排队导致写者不优先了,那就在读者外面外面加一层信号量。

同理,解决饥饿也只要加一层信号量!

- init\_semWF()初始化相关变量及信号量
- 读者进程基本流程WFreader()

```
P(&readerQueueSem);//防止一大群读者排队导致写者排队
if(solveHunger){
    P(&rHungSem);
}

//进大门
P(&readTry);
对readCount操作;
if(solveHunger){
    V(&rHungSem);
}
V(&readTry);//出大门

//读文件
P(&readersSem);
读文件;
V(&readersSem);
```

```
对readCount操作;
V(&readerQueueSem);
```

写者进程基本流程RFwriter()

```
if(solveHunger){
    P(&rHungSem);
P(&wMutex);
对wCount操作;
if(solveHunger){
   V(&rHungSem);
// 开始读啦
P(&resourceSem);
读文件;
V(&resourceSem);
对wCount操作;
```

### 截图

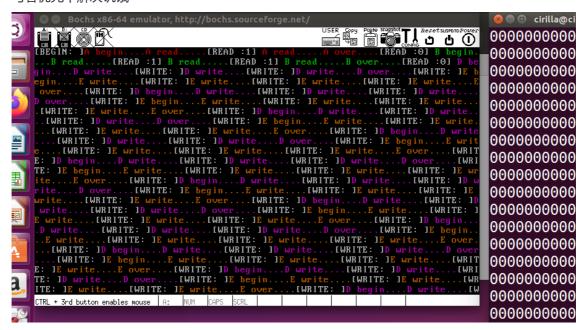
• 读者优先不解决饥饿

```
| Bochs x86-64 emulator, http://bochs.sourceforge.net/
| IBEGIN: JD begin...D write...[WRITE: 1D write...[URITE: 1D write...D over...
| IWRITE: JA begin...A read...[READ :1] B begin...B read...[READ :2] C begin...C read...[READ :3] A read...[READ :2] B read...[READ :2] C long...C read...[READ :2] A begin...A read...[READ :3] B read...B over...[READ :2] C read...C over...[READ :1] C read...C over...[READ :1] C begin...C read...[READ :2] B begin...B write...[WRITE: B write...[WRITE: B write...E over...[WRITE: B begin...B read...[READ :3] B read...[READ :2] A begin...A read...[READ :3] B read...[READ :3] C read...[READ :3] B read...B over...[READ :4] C over...[READ :4] D write...[WRITE: 1D write...[WRITE: 1D write...D over...[WRITE: 1D write...B over...[WRITE: 1D begin...B read...B over...[WRITE: 1D write...B over...[WRITE: 1D write
```

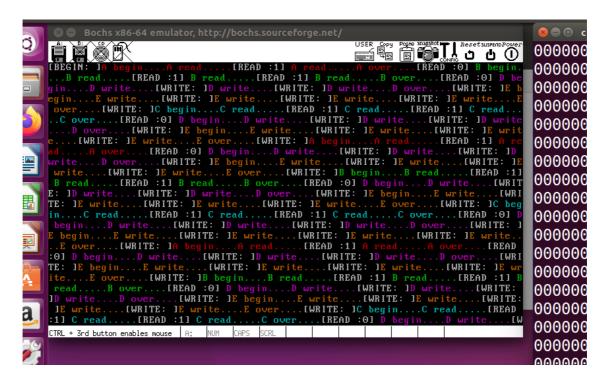
• 读者优先解决饥饿



• 写者优先不解决饥饿



• 写者优先解决饥饿



### 参考

[操作系统笔记——读者写者问题 - 掘金 (juejin.cn)] (https://juejin.cn/post/6844904041495937031)

[读者写者问题 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/189993251)