读者—写者问题的写者优先算法

王俊杰

(华中师范大学 计算机科学系,湖北 武汉 430079)

摘 要:分析了操作系统中读者-写者问题这一进程同步互斥问题,给出了两种信号量实现的写者优先算法。

关键词:信号量;写者优先;互斥;共享资源

中图分类号:TP312 文献标识码:A 文章编号:1672-7800(2008)02-0142-03

0 前言

在系统中,许多进程常常需要共享资源,而这些资源往往要求排它性的使用,因此进程间需要互斥地使用这些资源。用常规的程序来实现进程之间同步、互斥关系需要复杂的算法,而且还会造成“忙等待”,浪费CPU资源。

为此,著名的荷兰计算机科学家Dijkstra把互斥的关键含义抽象成信号量(Semaphore)概念,并引入在信号量上的P、V操作作为同步原语。

1 读者-写者问题及定者优先算法

读者-写者问题是一个信号量实现的经典的进程同步互斥问题。在系统中,很多竞争的进程试图读写一个数据集中的数据。多个进程同时读数据是可以接受的,但是如果一个进程正在写数据,而其它所有进程不能访问该数据,那么对于写者优先问题,就是要求读者写者同时访问时,写者优先。但是多数教材给出的是类似下面的程序。

各信号量的含义如下:

Wcount用于记录正在等待的写者的个数的整型变量,初值为0;

Rcount用于记录正在读的读者个数的整型变量,初值为0;

Wmutex:用于写计数的互斥信号量,初值为1;

Rmutex:用于读计数的互斥信号量,初值为1;

Read:用于写进程对读进程的互斥,初值为1;

Write:用于读进程对写进程互斥和写进程之间的互斥,初值为1。

写者:

```
begin
    p(wmutex);
    wcount := wcount +1;
    if(wcount==1)
        then p(read);
    v(wmutex);
    p(write);
    写数据集
    v(write);
    p(wmutex);
    wcount := wcount - 1 ;
    if(wcount==0)
        then v(read);
    v(wmutex);
end
```

读者:

```
begin
p(read);
p(rmutex);
rcount := rcount +1;
if(rcount==1)
    then p(write);
v(rmutex);
v(read);
读数据集
p(rmutex);
rcount := rcount - 1;
if(rcount==0)
    then v(write);
end
```

算法分析:假设有诸多读者R1,R2,R3……及诸多写者W1,W2,W3……都要访问数据集,对以下几种情况的优先、互斥情况分析如下:

(1)当写者先到达进入数据集访问,其先到达的次序为

作者简介:王俊杰(1987-),女,山东潍坊人,华中师范大学计算机科学系硕士研究生,研究方向为软件测试、操作系统进程。

W1、R1、R2、W2、W3、R3。因第一个写者W1访问数据集前要执行P(read),读者R1、R2、R3等待在信号量read上,第二、三个写者W2、W3等待在信号量write上,wcount=3。W1写完后唤醒写者W2、W3,写者得以先于读者访问;最后的写者访问完成后V(read)唤醒第一个读者R1,R1唤醒R2后去访问数据集,随后R3被唤醒,R2、R3可与R1同时访问数据集。

(2)当读者正访问数据集,又有新读者到来而无写者到来时,读者可共同访问数据集。

(3)当读者正访问数据集,又有诸多读者和写者到来时,次序为R1、W1、R2、R3、W2、W3,写者不能优于诸读者访问数据集。此时,第一个写者W1等待在信号量write上,因为读者R1已经执行V(read),第二、第三读者R2、R3可以进入临界区与R1共同访问数据集,第二、第三写者W2、W3也等待在信号量write上。只有当所以读者访问完释放write时,写者才可以逐个访问。

2　改进的写者优先算法

从上面的分析可以看到,当读者首先到来时,只有当所有读者全部读完,写者才可以执行。相同地,只有当写者先到达时,写者才会优于读者执行,体现出写者优先。

而写者优先算法要实现的是,首先要让读者与写者之间以及写者与写者之间互斥地访问数据集,其次,在无写进程到来时各读者可同时访问数据集。最重要的是,在读者和写者都等待访问时,写者优先。

下面是改进后的写者优先算法,各信号量的含义同前程序:

```
写者
begin
P(wmutex);
Wcount :=wcount +1;
If ( wcount == 1 )
        Then p(read);
V(wmutex);
P(write);
写数据集
P(wmutex);
Wcount := wcount - 1;
If ( wcount == 1 )
        Then V(read);
V(write);
V(wmutex);
end

读者
begin
P(wmutex);
V(wmutex);
P(rmutex);
Rcount:=rcount+1;;
If ( rcount == 1 )
        Then P(read)
V(rmutex);
读数据集
P(rmutex);
Rcount:=rcount- 1;
If ( rcount == 0 )
        Then P(read);
V(rmutex);
end
```

算法分析:假设有诸多读者R1,R2,R3……及诸多写者W1,W2,W3……都要访问数据集,对以下几种情况的优先、互斥情况分析如下:

(1)当写者先到达进入数据集访问,其先到达的次序为W1、R1、R2、W2、W3、R3。因第一个写者W1访问数据集前要执行P(read),第一个读者R1等待在信号量read上,其他的读者等待在信号量read上,第二、三个写者W2、W3等待在信号量write上,wcount=3。W1写完后唤醒写者W2、W3,写者得以先于读者访问;最后的写者访问完成后唤醒第一个读者R1,R1唤醒R2后去访问数据集,随后R3被唤醒,R2、R3可与R1同时访问数据集。

(2)当读者正访问数据集,有新读者到来而无写者到来时,读者可共同访问数据集。

(3)当读者正访问数据集又有诸多读者和写者到来时,次序为R1、W1、R2、R3、W2、W3,写者能优于诸读者访问数据集。此时,第一个写者W1等待在信号量read上,此时的第二、第三读者R2、R3不能进入临界区与R1共同访问数据集,而是等待在信号量wmutex上,第二、第三写者W2、W3也等待在信号量wmutex上。一旦R1完成,唤醒第一个写者W1去访问数据集,W1访问数据集之前它将唤醒R2,继而R3、W2、W3会被唤醒,R2重新被堵塞在信号量read上,R3被堵塞在信号量rmutex上,W2、W3会重新被堵塞在信号量write上。第一个写者W1访问完后,先唤醒写者W2、W3,最后一个写者完成后才唤醒读者R2、R3,因此写者W2、W3会优于读者R2、R3去访问数据集。

通过上面的分析,可以看到这种算法很好地实现了写者优先的要求,当然写者优先的算法不止一种,下面再给出一种从另一个角度出发的写者优先算法。

程序中用到的信号量表示的含义如下:

wwaitt用于记录正在等待的写者的个数的整型变量,初值为0;

Rwait用于记录正在等待的读者个数的整型变量,初值为0;

Rcount用于记录正在读的读者个数的整型变量,初值为0;

Mutex:用于修改计数值的互斥变量,初值为1;

Read:用于写进程对读进程的互斥,初值为0;

Write:用于读进程对写进程互斥和写进程之间的互斥,初值为0;

Sta表示当前状态是在读还是写，初值为NULL。

写者

```
begin
P(mutex);
If ( wwait>0 || sta == READING || sta == WRITNG)
{
    Wwait := wwait+1;
    V(mutex);
    P(write);
}
Else
    V(mutex);
```

写数据集

```
P(mutex);
If (wwait>0)
{
    Wwait:=wwait- 1;
    V(write);
}
Else
{
    If ( rwait>0)   sta = READING;
    For ( ; rwait>0; rwait- - )
    {
      Rcount:=rcount+1;
      V(read);
    }
}
V(mutex);
end
```

读者

```
begin
P(mutex)
If ( wwait>0 || sta == WRITING)
{
    Rwait:=rwait+1;
    V(mutex);
    P(read);
}
Else
{
    Rcount:=rcount+1;
    V(mutex);
}
```

读数据集

```
P(mutex)
Rcount- - ;
If( rcount == 0 && wwait>0 )
{
    Sta = WRITNG;
    Wait:=wait- 1;
    V(write);
}
V(mutex);
End
```

仍然按照上面的方法进行分析：假设有诸多读者R1,R2,R3……及诸多写者W1,W2,W3……都要访问数据集，对以下几种情况的优先、互斥情况分析如下：

（1）当写者先到达进入数据集访问，其到达的次序为W1、R1、R2、W2、W3、R3。第一个写者W1直接写数据集，当前状态为WRITING,读者R1、R2、R3执行If (wwait>0 || sta == WRITING), 堵塞在P(read)上，rwait=3。写者W2、W3执行If (wwait>0 || sta == READING || sta == WRITNG), 堵塞在P(write)上，wwait=2。W1写完后唤醒W2, wwait=1，继而唤醒W3, wwait=0.然后将状态置为READING, rcount=3,执行3个V(read),所以的读者可以执行。

（2）当读者正访问数据集又有新读者到来而无写者到来时，rcount增加，读者可共同访问数据集。

（3）当读者正访问数据集又有诸多读者和写者到来时，次序为R1、W1、R2、R3、W2、W3，写者能优于诸读者访问数据集。此时，第一个读者在读，rcount=1,写者W1、W2、W3等待在信号量P(write)上，wwait=3;读者R2、R3等待在P(read)上，rwait=2。R1读完后,rcount=0,执行If(rcount == 0 && wwait>0),释放write,W1开始执行，执行完后释放write,W2、W3相继执行。只有当W3执行完后，才会释放read, R2、R3才可以同时读数据集。

参考文献：

[1] 陈向群，杨芙清.操作系统教程[M].北京：北京大学出版社，2006.

[2] Andrew S.Tanenbaum.现代操作系统[M].陈向群，马洪兵译.北京：机械工业出版社，2006.

（责任编辑：刘 君）