一种改进的进程同步与互斥机制

罗 梅
（天津工程师范学院 计算机系，天津 300222）

摘 要：给出了利用一般信号量机制实现一类常见的进程互斥问题的方法，在分析该方法不足的基础上，提出了一种改进的进程同步与互斥机制——标签信号量。论述了标签信号量的设计、定义及其在实现进程同步与互斥中的应用。实例表明，在实现进程同步与互斥时，标签信号量比普通信号量更简单高效。

关键词：进程同步；进程互斥；一般信号量；标签信号量
中图分类号：TP274 **文献标识码**：A **文章编号**：1673－1018（2008）02－0040－04

An improved mechanism for process synchronization and mutual exclusion

LUO Mei
（Department of Computer Science，Tianjin University of Technology and Education，
Tianjin 300222，China）

Abstract：The method of realizing a common type of process′s mutual exclusion by using standard semaphore is given. An improved mechanism-tagged semaphore-is introduced based on the analysis of standard semaphore′s shortcomings. Tagged semaphore's design, definition and application in realizing process′s synchronization and mutual exclusion are discussed in detail. The examples show that tagged semaphore is more efficient than standard semaphore.

Key words：process′s synchronization; process′s mutual exclusion; standard semaphore; tagged semaphore

1965年由荷兰计算机科学家Dijkstra提出的信号量机制[1]是一种有效的用于实现进程同步与互斥的方法，它能消除进程的“忙等待”现象、操作简单且可实现任何类型的进程同步与互斥，因而成为研究并发程序行为的重要机制[2]，但是，信号量机制在实现某些类型的进程同步与互斥时十分复杂。如有这样一类进程互斥现象：某类多个资源任何时刻只能被同一类型的若干进程所访问，在部分该类资源被某些进程占用且还有部分剩余资源时，这部分剩余资源只能分配给与正在使用该类资源的进程相同类型的进程。本文利用一般信号量机制实现这类进程互斥，然后引入标签信号量机制并给出利用标签信号量实现这类进程互斥的方法。

1 一般信号量

1.1 一般信号量的数据结构及操作

荷兰科学家 Dijkstra 提出的一般信号量的数据结构及操作的定义为[3]：

```
Structure Semaphore
{int count;  //资源可用个数
pointer_PCB  queue;//等待该类资源的阻塞进程的PCB 队列
}
```

对一般信号量 s 的操作 Wait(s) 定义为：

```
{s.count--;
if(s.count<0) block(s, s.queue);
}
```

收稿日期：2007－11－09
作者简介：罗 梅（1974—），女，讲师，硕士，研究方向为软件工程.

对一般信号量 s 的操作 Signal(s)定义为：

```
{s.count++;
if(s.count<=0)wakeup(s,s.queue);
}
```

1.2　利用一般信号量实现进程互斥

1.2.1　问题描述

公用自习室问题：有一个公用自习室里有 *N* 个座位，规定任何时刻在自习室里的同学只能是同一个专业的。当自习室里无人时任何同学均可进去；当自习室里有同学且还有空座位时，与教室里的同学不同专业的同学不允许进去，但与自习室里的同学相同专业的同学可以进去。另如生产者与消费者使用公用缓冲池问题：任何时刻往缓冲池里存放消息的多个生产者进程提供的消息必须是同一类型的，如整数、字符串等，只有当缓冲池全部清空时才能往缓冲池里存放另一类型的消息。

1.2.2　问题分析

在公用自习室问题中，存在两类互斥。第一类是进入自习室的第一个同学与不同专业的同学之间的互斥；第二类是同一专业的同学之间由于座位数量有限而产生的互斥。当同学 *i* 要进入自习室时，先判断其中是否有人，如果没有则可以进去，同时设置专业标志；如果有人且还有空座位，则判断与自习室的同学是否同一专业，如果是则进去；否则（自习室满或专业不同），等待。假设只有两类专业，分析如下：

（1）由于在自习室里无人时任何同学都可以进去而在自习室里有人时只有同一专业的同学才可能进去，所以设一个共享变量 inCount 记录自习室里的同学人数，初值为 0。

（2）由于自习室里的同学的专业决定了后面可以进入自习室的同学的专业，所以设一个共享变量 inMajor 标志自习室里的同学的专业，初值为空。

（3）如果有另一个专业的多个同学在等待进入自习室，则当自习室清空时可以连续让多个同学进去，所以设一个共享变量 outCount 记录等待的同学人数，初值为 0；设一个共享变量 outMajor 记录等待的同学的专业，初值为空。

（4）对于所有同学来说，变量 inCount、outCount、inMajor 和 outMajor 必须互斥访问，所以设置一个互斥信号量 mutexR 表示对它们的互斥操作，初值为 1。

（5）由于第一个进入自习室和最后一个退出自习室的同学决定了其它专业的同学是否可以进去，所以设置一个互斥信号量 mutex 表示“是否允许进”，初值为 1。

（6）对于同一专业的同学来说，座位数量是有限的，所以设置一个互斥信号量 mutex*N* 表示对座位的互斥操作，初值为 *N*。

1.2.3　问题解决

```
//定义信号量及公用变量
Semaphore mutexR={1,Null},mutex={1,Null},
mutexN={N,Null};
int inCount=0,outCount=0,i=0;string inMajor="",
outMajor="";
//设同学 i 的活动对应于进程 Process i,则进程
Process i 的代码如下：
Process i:
{Wait(mutexR);
 if(inCount==0){inMajor=同学 i 的专业;
Wait(mutex);
                inCount++;Signal(mutexR);
Wait(mutexN);}
 else if(inMajor==同学 i 的专业){
                inCount++;Signal(mutexR);
Wait(mutexN);}
 else{outCount++;outMajor=同学 i 的专业;
      Signal(mutexR);Wait(mutex);
      inCount++;Wait(mutexN);}
 在自习室里自习;
 Wait(mutexR);
 inCount--;
 Signal(mutexN);
 if(inCount==0){inMajor=outMajor;outMajor="";
                for(i=1;i<=outCount;i++)
                Signal(mutex);}
 Signal(mutexR);
 }
```

一般情况下，在用一般信号量实现这类进程互斥时需引入共享变量，并成为一类新的互斥资源，所以在进程的代码中引入三个互斥信号量以实现三类互斥。在用信号量实现进程同步与互斥时，信号量的 Wait 操作和 Signal 操作必须成对出现。由于本问题的复杂性，上述代码中的 Wait 操作和 Signal 操作的成对出现不在文字上表现出来，而是通过添加一个循环结构来控制。由此可见，在利用一般信号量实现复杂的进程同步与互斥时申请与释放资源的操作很复杂、易出错，而且当进程的类型更加多样（如公用自习室问题

中同学的专业类型不只两种而是有更多种)时,实现也更加复杂。为了简单高效地实现这类进程同步与互斥,可对一般信号量进行改进,从而引入标签信号量的概念。

2 标签信号量

2.1 标签信号量的设计

在上述进程互斥问题中,由于资源的申请与释放操作的发生不仅仅与资源可用个数有关,还与正在使用部分该类资源的进程的类型有关,进程在申请资源时必须提供其类型,就像是出示工作证一样,而对资源的管理不仅仅要维护其可用个数,还要记录正使用该类资源的进程类型,所以对一般信号量进行扩充,引入一个标签分量 Tag,用于标识正在使用资源的进程的类型,当所有资源均可用时,该标签的值为空;一般信号量原来的两个分量 count 和 queue 在标签信号量中仍旧保留,count 意义不变,但在标签信号量中,阻塞进程的 PCB 和类型值均应参与排队,以便于释放资源的进程根据阻塞进程的类型选择要唤醒的进程。标签信号量也是一个被保护的变量,操作原语为 TWait 和 Tsignal。TWait 用于申请资源,有两个参数:一个是标签信号量 s;另一个是 tag,表示申请资源的进程的类型。TSignal 用于释放资源,由于进程类型不影响其释放资源,所以该原语只有一个参数, 即标签信号量 s。

2.2 标签信号量的数据结构及操作

根据上述设计思想,标签信号量的数据结构及操作可定义为[4~6]:

```
Structure Tagged-Semaphore
{int count;  //资源可用个数,初值为该类资源个数 N
string Tag;//正使用该类资源的进程的类型标签,该类资源全部可用时该值为空
pointer_PCB_Tag queue;//等待资源的阻塞进程队列,进程的 PCB 和类型值参与排队
}
```

其中,pointer_PCB_Tag 的数据结构定义为:

```
Structure Pointer_PCB{PCB pcb;//进程的 PCB 地址
                      String tag;//进程的类型
                      }
```

对标签信号量 s 的操作 TWait(s, tag)定义为:

```
{if(s.count==N){s.count--;  s.Tag=tag;}
else if((s.count>0)&(s.Tag==tag))s.count--;
else block(s,tag,s.queue);//进程阻塞,其类型标签 tag 也参与排队
}
```

对标签信号量 s 的操作 TSignal(s)定义为:

```
{if(s.queue 非空){
if(s.count==N-1)  //该进程是最后一个使用该类资源的进程
{wakeup(s.queue);//唤醒一个等待的进程并取出其结点 node
s.Tag=node.tag;//将 s.Tag 置为被唤醒的进程的 tag 值
while((s.count>0)&(s.queue 中仍有标签为 s.Tag 的进程))
        {wakeup(s.queue);  s.count--;}
}
else if((s.count>0)and(s.queue 中有标签为 s.Tag 的进程))
        wakeup(s.queue);
}
else  s.count++;
}
```

2.3 利用标签信号量实现进程互斥

利用标签信号量很容易解决前述的公用自习室问题,问题描述与分析同前。设同学 i 的活动对应于进程 Process i,则可将进程 Process i 做如下描述:

```
Tagged-Semaphore s={N,"",Null};
Process i:
tag=major;//将其标签设置为其专业
TWait(s,tag);
在自习室里自习;
TSignal(s);
```

在利用标签信号量实现进程互斥时,其 TWait 与 TSignal 操作与一般信号量的 Wait 与 Signal 操作一样,应成对出现在同一进程代码中。显然利用标签信号量实现进程互斥的效率显著提高,而且大大降低了出错率。其原因是标签信号量的原语操作已经接管了复杂的资源申请与释放操作。

3 标签信号量的应用

标签信号量不仅可以高效地实现进程互斥,而且可以实现进程同步。如生产者-消费者问题:多个提供不同类型消息的生产者和多个消费者共享有 N 个缓冲区的缓冲池,要求缓冲池中不能同时存放不同类型的消息且任何时刻只能有一个进程访问缓冲池。当缓冲池中存放有某种类型的消

息时，只要还有空缓冲区，提供同一类型消息的生产者就可以往缓冲区中存放消息而生产其它类型消息的生产者就只能等待缓冲池清空。消费者进程不断地从缓冲池中取出消息。在这个问题中，进程之间既有同步关系又有互斥关系，可以把一般信号量与标签信号量结合起来实现。设置一般信号量 Mutex 用于缓冲池的互斥访问，设置标签信号量 Empty 表示空缓冲区数，设置一般信号量 Full 表示满缓冲区数。若生产者进程 i 用 Producer i 表示，可将其描述如下：

```
Semaphore Mutex={1, Null}, Full={0, Null};
Tagged-Semaphore Empty={N, "", Null};
process Produceri
{生产一个消息 m;  //消息名称
 tag=消息 m 的类型;  //消息类型
 TWait(s, tag);//申请一个空缓冲区，须检验其类型
 Wait(Mutex);//互斥访问缓冲池
 将消息存入缓冲区;
 Signal(Mutex);//释放缓冲区访问权
 Signal(Full);//通知消费者又多一个满缓冲区
}
```

若消费者进程 j 用 Consumerj 表示，描述为：

```
Process Consumerj
{Wait(Full);
 Wait(Mutex);//互斥访问缓冲池
 从缓冲区中取出消息;
 Signal(Mutex);//释放缓冲区访问权
 TSignal(Empty);//通知生产者又多一个空缓冲区
}
```

在利用标签信号量实现进程同步时，其 TWait 与 TSignal 操作与一般信号量的 Wait 与 Signal 操作一样，应成对出现在有同步关系的不同进程代码中。

4　结束语

标签信号量虽然比一般信号量多了一个分量，且用于申请资源和释放资源的两个操作原语的定义复杂了，但是通过两种机制的应用实例可以得出：标签信号量的描述能力比一般信号量强得多，所以标签信号量机制对于实现复杂的进程同步与互斥来说是相当有意义的。引入标签信号量机制以后，在系统中可以不保留一般信号量机制，如果不需要对进程进行分类，则可以忽略标签信号量的第二个分量 Tag，这时标签信号量退化为一般信号量。也可以同时使用一般信号量和标签信号量，将一般信号量与标签信号量结合起来实现进程的同步与互斥，这对于操作系统来说只不过是多了两个操作原语，而其实现并行程序设计的能力却提高了很多。当然，本文介绍的标签信号量机制的描述能力仍不是最强的，在实现多个公用自习室的互斥访问时依然相当复杂，所以仍有改进与扩展的空间。

参考文献：

[1] DIJKSTRA E W. The structure of the “THE” multiprogramming system [J]. Comm of ACM, 1968, 11(5): 341 – 346.

[2] ANDREWS G R. Concurrent Programming: Principles and Practice [M]. Benjamin: Cummings Publishing Company Inc, 1991.

[3] 屠　祁, 屠立德. 操作系统基础 [M]. 北京：清华大学出版社, 2002.

[4] 徐宝文. 带标记信号量——一种新型同步与互斥机制 [J]. 计算机科学, 2001, 28(1): 15 – 17.

[5] 曲维光. 带标记信号量机制的拓广 [J]. 小型微型计算机系统, 2001, 22(12): 1526 – 1528.

[6] 彭迎春, 谭汉松. 带标记数组型信号量机制的扩展 [J]. 深圳信息职业技术学院学报, 2005, 3(3): 17 – 20.