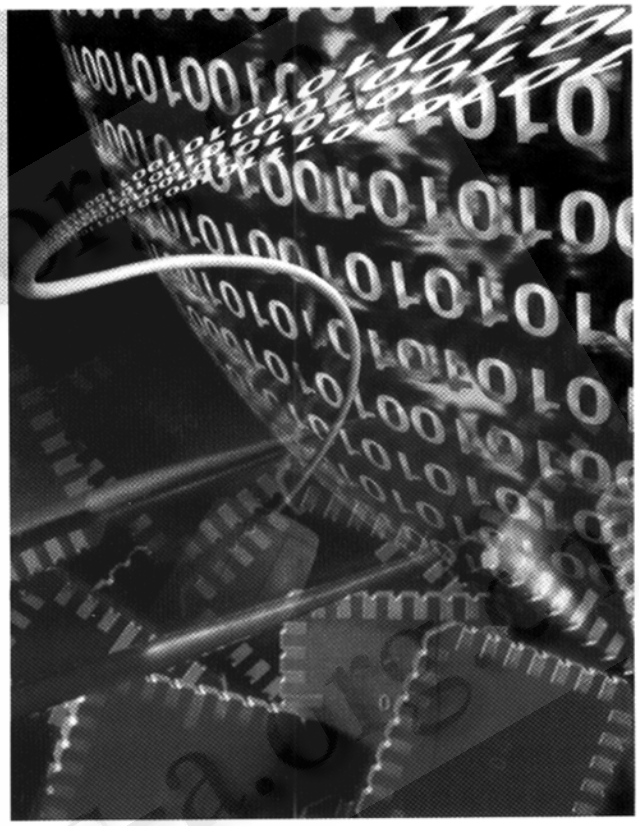
多线程应用程序中的同步控制技术及应用

于华 （山东财政学院计算机信息工程系 250014）

摘要：设计多线程应用程序必须在线程之间保持一定的同步关系，才能使用户能够对独立运行的线程进行有效的控制，以保证线程的安全运行。这是多线程编程中最关键也是最复杂的问题。本文将介绍多线程间的同步控制方法，并给出了在Delphi中的应用实例。

关键词：线程 同步对象 等待函数



1 引言

在Windows 95/98/NT中所有的应用程序实际上都以是线程的方式运行的。由于同一进程的所有线程共享进程的虚拟地址空间，并且线程的中断是汇编语言级的，所以可能会发生两个线程同时访问同一个对象（包括全局变量、共享资源、API函数等）的情况，这有可能导致程序错误。例如，一个线程正在更新一个结构的内容的同时，另一个线程正试图读取该结构。结果，我们将无法得知所读取的数据是什么状态。

属于不同进程的线程在同时访问同一内存区域或共享资源时，也会存在同样的问题。因此，在多线程应用程序中，常常需要采取一些措施来同步线程的执行。Win 32 API提供了同步对象、等待函数等同步机制来解决这个问题，本文将介绍这些同步机制的使用方法，并给出了在Delphi中的应用实例。

2 等待函数

Win32 API提供了一组能使线程阻塞其自身执行的等待函数，这些函数只有在作为其参数的一个或多个同步对象(见下文所述)产生信号时才会返回。在超过规定的等待时间后，不管有无信号，函数也都会返回。在等待函数未返回时，线程处于等待状态，此时线程只消耗很少的CPU时间。

使用等待函数即可以保证线程的同步，又可以提高程序的运行效率。最常用的等待函数是WaitForSingleObject，该函数的声明为：

DWORD WaitForSingleObject(HANDLE hHandle, DWORD dwMilliseconds);

参数hHandle是同步对象的句柄。参数dwMilliseconds是以毫秒为单位的超时间隔，如果该参数为0，那么函数就测试同步对象的状态并立即返回，如果该参数为INFINITE，则超时间隔是无限的。函数的返回值及其含义如下：

WAIT_OBJECT_0 指定的同步对象处于有信号状态

WAIT_TIMEOUT 超时返回，并且同步对象无信号

WAIT_FAILED 函数失败

WAIT_ABANDONED 拥有一个mutex的线程已经中断了，但未释放该MUTEX

3 同步对象

同步对象用来协调多线程的执行，它可以被多个线程

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

共享。同步对象的状态要么是有信号的，它使等待函数返回，要么是无信号的，它禁止等待函数返回。Win 32 API主要提供了三种同步对象：mutex、信号灯和事件。

3.1 事件对象

事件对象(Event)是最简单的同步对象，当它有信号时，所有等待它的线程都可以通过，当它无信号时，所有等待它的线程都会被阻塞。事件对象通常用于在应用程序访问某一资源之前应用程序必须等待的情况(比如，在数据写进一个文件之前数据必须从通信端口得到)。

可以用 Win32 API 的 CreateEvent 函数来建立事件对象。但 Delphi 用 TEvent 对象封装了该对象，使该对象的使用更加简单和方便。首先创建一个全局的 TEvent 对象作为所有线程可监测的标志。当一个线程完成某项特定的操作时，调用 TEvent 对象的 SetEvent 方法，这样将设置这个标志，其他的线程可以通过监测这个标志获知操作的完成。相反，要取消这个标志，可以调用ResetEvent方法。

3.2 互斥对象

mutex 对象具有排它的拥有属性，用于保证一个资源一次只能有一个线程访问。其状态在它不被任何线程拥有时是有信号的，而当它被拥有时则是无信号的。mutex 对象很适合用来协调多个线程对共享资源的互斥访问(mutually exclusive)，每个线程在它执行访问共享资源的代码之前必须等待Mutex的所有权。例如，有几个线程共享对数据库的访问时，线程可以使用 mutex 对象，每次只允许一个线程向数据库写入。

线程用 CreateMutex 函数来建立 mutex 对象。该函数的声明如下：

```
HANDLE CreateMutex(
  LPSECURITY_ATTRIBUTES lpMutexAttributes,  // 安全属性
  BOOL bInitialOwner, // 初始是否拥有
  LPCTSTR lpName     // 为对象指定名字,其他进程中的线程可通过该名字获得 mutex 对象的句柄
);
```

在完成对共享资源的访问后，线程可以调用ReleaseMutex 来释放 mutex，以便让别的线程能访问共享资源。如果线程终止而不释放 mutex，则认为该 mutex 被废弃。

3.3 信号灯对象

信号灯对象维护一个0到指定最大值之间的计数，其状态在计数值大于0时是有信号的，而在计数值为0时则是无信号的。信号灯对象可用来限制对共享资源进行访问的线程数量。例如，应用程序要限制它建立的窗口数。线程用 CreateSemaphore 函数来建立信号灯对象，该函数的声明如下：

```
HANDLE CreateSemaphore(
  LPSECURITY_ATTRIBUTES lpSemaphoreAttributes,  // 安全属性
  LONG lInitialCount, // 对象的初始计数
  LONG lMaximumCount, // 最大计数
  LPCTSTR lpName     // 为对象指定名字
);
```

一般把信号灯的初始计数设置成最大值。每次当信号灯有信号使等待函数返回时，信号灯计数就会减1，而调用ReleaseSemaphore可以以指定量增加信号灯的计数。计数值越小就表明访问共享资源的程序越多。

信号灯其实是互斥的一种推广。即它可以被n个线程所共同占有，而多于 n 个的等待线程则会被阻塞。

互斥和信号灯Delphi都没有提供其封装对象，因此在Delphi程序中要使用它们必须借助于Win 32 API函数。下面所述实例中提供了这几个对象的使用方法。

4 应用实例

本例是对一个实际的《电能表远程抄表系统》中电话拨号抄表和将电表数据写入数据库这一过程的模拟。由于这两个操作都比较费时，为了减少用户的等待时间，提高应用程序的效率，程序中引入了多线程技术，分别用两个线程完成电话拨号抄表和写入数据库这两个操作。抄表线程从电表采集数据后先放入缓冲区，写数据库线程从缓冲区读取数据写入数据库。为了防止两个线程同时对缓冲区进行操作，在两个线程中使用了互斥对象；为了限制抄表线程向缓冲区写入太多的数据，使用了信号灯对象以控制向缓冲区写入的数据量，每当抄表线程向缓冲区写入一次数据后，信号灯计数加1，而当写数据库线程读取一次数据后，信号灯计数减1；另外程序中还创建了两个事件对象：EvThreadLoop、EvCanCommand，用以实现主线程和抄表线程的同步。抄表线程的开始，首先等待事件evThreadLoop的信号状态，如果evThreadLoop有信号，自动将它置为无信号，进入循环，执行操作，操作执行完毕后，置 EvCanCommand 为有信号，允许主线程发布抄表或其它操作，然后进入回到循环开始，等待事件

(下转第 52 页)

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn

(上接第 48 页)

evThreadLoop 的状态。主线程在发布抄表命令时，首先要等待事件evCanCommand的状态，如果evCanCommand无信号，表示抄表线程正在进行执行一些操作，该函数调用退出，如果 evCanCommand 有信号，表示抄表操作结束，可以开始其他操作，主线程就指定操作，置 evThreadLoop 为有信号，抄表线程自动启动。程序代码略。

模拟主窗体界面如下。

主要事件过程：

```
procedure TfrmMain.FormCreate(Sender: TObject);
begin
hSemCanGet:=CreateSemaphore(nil, 0, MaxSemPooler, nil);// 创建信号灯对象，抄表线程与写入数据库线程同步用，抄表线程把数据写入 RecvDataList 后把 hSemCanGet 信号量加 1。
hSemCanWrite := CreateSemaphore(nil, MaxSemPooler, MaxSemPooler, nil);// 信号灯对象 抄表线程与写入数据库线程同步用，写数据库线程从RecvDataList中取出数据后把 hSemCanWrite 信号量加 1。
HMutex:=CreateMutex(nil,False,nil)// 创建互斥对象，初始不取得所有权
RecvDataList := TList.Create;
RecvSet := TRecvSet.Create(False); // 创建抄表线程
WriteData := TWriteData.Create(False);// 创建写数据库线程
end;
procedure TfrmMain.ToolButton1Click(Sender: TObject); // 抄表按扭的 Click 事件
begin
RecvSet.Operate(1);
end;
procedure TfrmMain.ToolButton2Click(Sender: TObject);// 设置按扭的 Click
begin
RecvSet.Operate(2);
end; ■
```


参考文献

1 徐新华 编著，GUI 编程技术，人民邮电出版社，1998。

2 Microsoft Corporation 著，Win32 程序员参考大全，清华大学出版社，1995。

©《计算机系统应用》编辑部 http://www.c-s-a.org.cn