扩展Delphi的线程同步对象

于 华 (山东财政学院计算机信息工程系 250014)

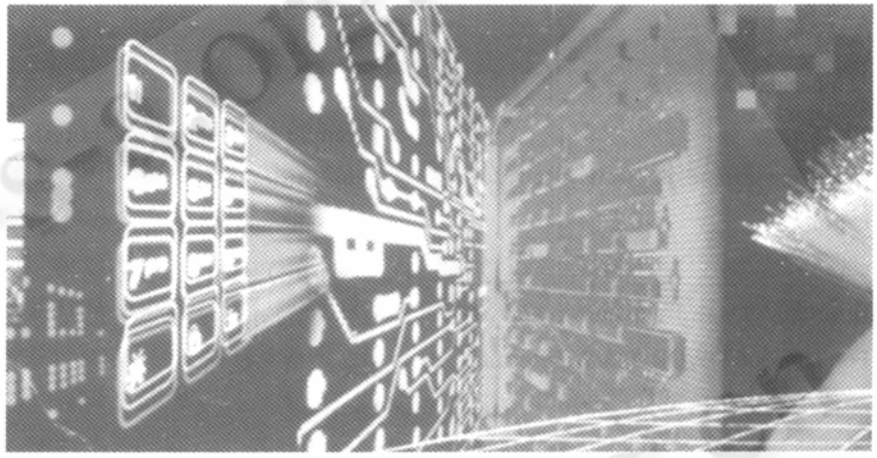
摘要：Delphi只提供了事件（TEvent）和临界区（Tcritical）两个同步对象来控制线程间的同步资源访问，如果要使用其他的同步对象如：信号灯、互斥等，就必须借助于Win 32 API函数，颇感不便，因此，本文用Delphi构造了两个类对信号灯和互斥对象进行了封装（分别为Tsemaphore和Tmutex），以期对广大Delphi编程人员有所帮助。

关键词：Delphi 线程 同步对象

编写多线程应用程序最重要的是控制好线程间的同步资源访问，以保证线程的安全运行。Win 32 API提供了一组同步对象，如：信号灯(Semaphore)、互斥（Mutex）、事件(Event)、临界区（CriticalSection）等来解决这个问题。Delphi分别将事件对象和临界区对象封装为Tevent对象和TcritialSection对象，使得这两个对象的使用简单、方便。但是如果在Delphi程序中要使用信号灯或互斥等对象就必须借助于繁杂的Win 32 API函数，很不方便，尤其是对那些不熟悉Win 32 API函数的编程人员来说，更是如此。因此，本文用Delphi构造了两个类对信号灯和互斥对象进行了封装(分别为Tsemaphore和Tmutex)，以期对广大Delphi编程人员有所帮助。

1 类的构造

先对Win32 API的信号灯对象和互斥对象进行抽象，构造一个父类ThandleObjectEx，在父类中实现了一个等待函数WaitFor和一个Release方法。等待函数用以判定信号灯或互斥对象是否是有信号的，若同步对象有信号，则等待函数返回，否则线程处于等待状态。Release方法用以释放同步对象。然后由这个父类派生出



两个子类Tsemphore和Tmutex。在子类Tmutex中重载了其父类的Release方法，在子类Tsemphore中超载了其父类的Release方法。

类的源代码如下：

```
unit SyncobjsEx;
interface
uses Windows, Messages, SysUtils, Classes,Syncobjs;
type
  THandleObjectEx = class(THandleObject)// ThandleObjectEx为互斥类和信号灯类的父类
  protected
    FHandle: THandle;
    FLastError: Integer;
public
    destructor Destroy; override;
    procedure Release;override;
      function WaitFor(Timeout: DWORD): TWaitResult;
    property LastError: Integer read FLastError;
    property Handle: THandle read FHandle;
  end;
  TMutex = class(THandleObjectEx)// 互斥类
  public
     constructor Create(MutexAttributes: PSecurityAttributes;
      InitialOwner: Boolean; const Name: string);
    procedure Release; override;
  end;
    TSemaphore = class(THandleObjectEx)// 信号灯类
    public
   constructor Create(SemaphoreAttributes: PSecurityAttributes;
```

```
        InitialCount: Integer; Maxim-
umCount: integer; const Name: string);
    procedure Release(ReleaseCount:
Integer=1;
        PreviousCount: Pointer=nil);
overload;
  end;
implementation
{ THandleObjectEx }// 父类的实现
destructor THandleObjectEx.Destroy;
begin
  Windows.CloseHandle(FHandle);
  inherited Destroy;
end;
procedure THandleObjectEx.Release;
begin
end;
function THandleObjectEx.Wait-
For(Timeout: DWORD): TWaitResult;//
等待函数，参数为
等待时间
{等待函数只有在作为其参数的一
个或多个同步对象(如信号灯、互斥
等)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。}
begin
  case WaitForSingleObject(Handle,
Timeout) of
    WAIT_ABANDONED: Result :=
wrAbandoned;// 无信号
    WAIT_OBJECT_0: Result :=
wrSignaled;// 有信号
    WAIT_TIMEOUT: Result :=
wrTimeout;// 超时
    WAIT_FAILED:// 失败
    begin
      Result := wrError;
      FLastError := GetLastError;
    end;
  else
    Result := wrError;
  end;
end;
{ TSemaphore }// 信号灯类的实现
constructor TSemaphore.Create(Se-
maphoreAttributes: PSecurity-
Attributes;
    InitialCount, MaximumCount:
integer; const Name: string);// 信号灯
类的构造函数
begin
  FHandle := CreateSemaphore(Se-
maphoreAttributes,
    InitialCount, MaximumCount,
PChar(Name));// 四个参数分别为：安
全属性、初始信号灯记数、最大信号
灯记数、信号灯名字
end;
procedure TSemaphore.Release(Re-
leaseCount: Integer=1;
  PreviousCount: Pointer=nil); 信号灯
类的 Release 方法，每执行一次按指
定量增加信号灯记数
begin
  Windows.ReleaseSemaphore(FHandle,
ReleaseCount, PreviousCount);
end;
{ TMutex }// 互斥类的实现
constructor TMutex.Create(Mu-
texAttributes: PSecurityAttributes;
  InitialOwner: Boolean; const Name:
string);// 互斥类的构造函数
begin
  FHandle := CreateMutex(Mutex-
Attributes, InitialOwner, PChar(Name));
end;
procedure TMutex.Release;//互斥类的
Release 方法，用来释放对互斥对象
的所有权
begin
  Windows.ReleaseMutex(FHandle);
end;
end.
```

2 信号灯对象与互斥对象的使用

2.1 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，在其计数大于0时是有信号的，而在其计数为0时是无信号的。信号灯对象可用来限制对共享资源进行访问的线程数量，如应用程序可使用信号灯对象限制它建立的窗口数量。

用类的 Create 方法来建立信号灯对象，在调用该方法时，可以指定对象的初始计数和最大计数。该方法有四个参数，依次为：安全属性、初始计数、最大计数、对象名字（以便别的进程的线程可打开指定名字的信号灯句柄）。如：

```
Semaphore := TSemaphore.Crea-
te(nil,10,10,'');
```

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，而通过调用对象的 Release 方法可以以指定量增加信号灯的计数(默认为加1)。计数值越小就表明访问共享资源的程序越多。如：Semaphore.Re-lease(3, nil);//第一个参数为增加的信号灯数量，第二个参数为执行该方法之前的信号灯数量。信号灯用法举例：

在线程建立窗口之前，它使用 WaitFor 函数确定信号灯的当前记数是否允许建立新的窗口，等待时间设为 10 秒。

(下转第 46 页)

(上接第 40 页)

```
if wrSignaled = Semaphore.Wait-For(10000) then// 若信号灯是有信号的
  begin
    //打开另一个窗口
  end
```

当线程关闭窗口时，调用 Release 方法递增信号灯计数

```
Semaphore.Release()
```

2.2 互斥对象

mutex对象的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)。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，一次只允许一个线程向数据库写入。

用类的Create方法建立mutex对象，在建立mutex时，可以为对象起个名字，这样其他进程中的线程可以打开指定名字的 mutex 对象句柄。如：

```
Mutex := TMutex.Create(nil, False, '');//第一个参数为安全属性，第二个参数为初始状态（True 为初始拥有，False 为初始不拥有），第三个参数可为互斥对象指定名字。
```

在完成对共享资源的访问后，可以调用Release 方法来释放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。

互斥对象用法举例：

```
if wrSignaled = Mutex.WaitFor(10000) then//若获得互斥对象的拥有权
  begin
    try
      //往数据库写入
    finally
      Mutex.Release;// 释放对互斥对象的拥有权
    end;
  end;
```

上述类代码在Windows98，Delphi5.0下调试通过，并通过了正确性测试。■

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*