实时动态模拟环境中多任务间的数据共享

陈文宇* 何元清

（电子科技大学计算机科学与工程学院 成都 610054）（中国民航飞行学院基础部 四川广汉 618300）

【摘要】 根据不同应用程序之间有共享数据的要求,针对 Windows NT 系统和 VC++的特点,提出通过动态链接库提供共享数据的方法,解决了 Windows NT 系统下实时动态模拟环境中多任务间数据共享，结果使各应用程序共同使用一个动态链接库，而需要共享的数据在该库中定义，达到了共享的目的。且也使不同语言的应用程序之间达到共享数据。通过地址访问方式还可以使各个应用程序通过间接访问方式使用共享数据。该方法简便、实用性强。

关 键 词 共享变量； 动态链接库； 多任务； 变量访问方式

中图分类号 TP319

WindowsNT 系统下实时动态模拟环境是一个基于 WindowsNT 系统下的实时多任务应用软件的运行平台。该软件通过使用 NT 系统的现有资源，在主模拟计算机中实现多任务的建立、调度管理、删除以及各任务间的数据共享，通过以太网实现和外围计算机的实时数据交换，通过用户可以自定义的图形用户界面观察整个系统的运行情况。整个软件系统功能可以分为以下几个部分：多任务的建立、调度管理、删除；多任务间的数据共享；以太网的实时通信；用户可以自定义的图形用户界面。

本文介绍的是多任务间的数据共享的实现方法。

1 数据共享的几种方法

共享数据可以通过很多方式进行，如通过磁盘文件或数据库，将共享数据存放在一个磁盘文件或数据库中，通过对该文件或数据库的的读写，达到数据共享；或通过共享内存的方式，共享内存的方法可以归纳为几种：内存映像文件、共享内存页(又分为动态申请和静态申请两种方式)、定制资源；内存映像文件是其他内存共享方法的基础。

内存映像文件：内存映像文件 I/O 是 Win9x API 处理磁盘文件的一种方式。内存映像文件 I/O 允许两个或多个进程共享基于文件的数据。每个和共享有关的进程直接存取一组公共页。由于共享占用的资源最小，当大量的数据必须被共享时，通过内存映像文件 I/O 共享就显得非常有用。需要使用 API 提供的 CreateFile( )、OpenFile( )、CreateFileMapping( )、OpenFileMapping( )、DuplicateHandle( )、MapViewOfFile( )等函数。

共享内存页中的动态申请共享页：调用 CreateFileMapping( )创建一个文件映像对象，并通过文件尺寸参数设定共享内存区域的大小，文件映像对象将自己连接到系统页文件，然后调用 MapViewOfFile( )创建内存视图，以确定要访问的特定内存区域，包括起始点和大小。

共享内存页中的静态分配共享页：Microsoft Visual C++编译器提供在不同进程间共享全局变量的能力。编译和连接程序在执行文件中分配共享页，在运行时刻，加载程序允许这些页面同时映射到几个进程的地址空间。

定制资源：定制资源像其他资源一样被编在资源文件中，并被链接程序捆绑到执行文件中任何大段的只读数据都可用做定制资源。用定制资源方法保存只读数据有以下优越性：由于数据被捆绑在执行文件中，查询数据很安全；提高内存使用效率；简化编程任务。

2001 年 6 月 20 日收稿

* 男 32 岁 硕士 讲师

虽然共享数据的两类方式可以提供数据共享，但共享文件或数据库方式不利于实时性要求，对共享数据的安全也不能保证；而共享内存方式不能使不同语言的应用程序之间共享数据。通过动态链接库共享数据的方法，是在静态的共享内存页的基础之上，利用动态链接库定义共享数据的方式，用户可以不必考虑共享页面的分配和管理，是一种简便、实用的实时共享数据的方式，并且也能使不同语言的应用程序之间共享数据[1,2]。

2 动态链接库

动态链接库在 Windows 应用程序中使用动态链接库有很多的好处，最主要的一点是它可以使得多个应用程序共享一段代码，从而可以大幅度的降低应用程序的资源开销，同时缩小了应用程序的最终执行代码的大小。此外，通过使用动态链接库，可以把一些常规的例程独立出来，有效的避免了不必要的重复开发，并且，由于应用程序使用了动态链接的方式，还可以在不需重新改写甚至编译应用程序的基础上更新应用程序的某些组件[3]。

3 WindowsNT 系统下实时动态模拟环境中的多任务

模拟环境下的实时多任务，是指在 NT 系统下通过进程实现的、具有不同的优先级的、可以独立运行的、用户开发的应用程序。各任务之间通过共享内存的方式实现数据交换，以保证软件系统运行的实时性。需要注意的是，此处涉及到的共享内存是指各任务之间进行数据交换的一段存储空间，而不是指多个 CPU 之间的共享内存。

各个多任务需要单独编译，在每个任务中可以通过直接访问变量(使用变量名)的方式或者通过变量的名称（使用变量的名字—字符串）的方式来访问共享的数据。

3.1 WindowsNT 系统下实时动态模拟环境中的多任务共享数据

采取静态分配共享页和动态链接库的方式，为了共享全局变量，须要满足初始化全局变量、声明要共享的全局变量、给执行程序中包含共享全局变量的部分赋予共享属性等三个条件。共享数据 DLL 允许进程以共享数据的方式访问读写数据，多个进程都可以对该共享数据 DLL 进行数据操作，达到共享数据的目的。建立共享内存，必须执行以下步骤：首先创建一个有名的数据区，在 Visual C++中是使用 data_seg pragma 宏。使用 data_seg pragma 宏必须注意数据的初始化；然后在用户的 DEF 文件中为有名的数据区设定共享属性；最后在应用程序(进程)按外部变量引用共享数据。将各个任务所需要共享的数据在一个动态链接库中定义，为保证任务能共享数据，使用的应该是同一个动态链接库，那么各个任务能够直接使用变量名来访问共享数据；为了也能通过变量名称的方式来访问共享的数据，动态链接库中提供外部函数供各个任务调用，该函数使用绝对地址的方式(即使用指针变量方式)来提供访问，既将指针变量本身也作为共享数据，各个任务直接使用共享的指针数据再访问真正需要的数据[4, 5]。

3.2 相关的文件和初始化共享数据的程序

为了方便共享数据的修改，包括增加、删除和修改共享数据名，把共享数据存放在一个数据文件中， 需要先定义的数据文件有：

var_define.h　存放变量的定义(为了变量能够共享，每个变量都必须要初始化)

注意：共享数据不能包括有指针类型的数据变量。

初始化共享数据的程序，创建必需的文件，它们是：

pointer_define.h

定义共享变量和存放共享变量实际地址的指针数组文件，在每个共享变量的定义(变量需要初始化)前加上 SHARED_VAR_API 的说明；

pointer_extern.h

共享变量和存放共享变量实际地址的指针数组的文件，在每个共享变量的声明(变量不需要初始化)前加上 extern SHARED_VAR_API 的说明；

var_num.h　定义共享变量数量(动态链接库 SHARED_VAR 需要)

3.3 动态链接库 SHARED_VAR

动态链接库提供共享变量的共享定义和通过变量的名称(使用变量的名字—字符串)的方式来访问共享的数据的函数。

提供共享变量的共享定义;

SHARED_VAR.cpp 中应该有：

```
#pragma data_seg(".varshared")
#include "pointer_define.h"
#include "struct_shared.h"
#pragma data_seg()
#pragma comment(linker,"/section:.varshared,RWS")
```

SHARED_VAR.h 中应该有：

```
#include "pointer_extern.h"
```

提供 Get_Var_Value(char * var_name)函数，以便 VC++的应用程序可以通过变量名字来取得共享变量的值(通过变量的实际地址来实现)。

多个任务使用动态链接库

各个应用程序需要使用到的动态链接库，必须将该动态链接库的*.lib 和*.h 文件(*代表动态链接库的名字)拷贝到应用程序的文件夹中。

3.4 各个应用程序应该包含的文件

各个应用程序应该包含的文件为：

```
#include "SHARED_VAR.h"
#include "Read_Write_File.h"
```

3.5 设置链接环境

应用程序如在 VC++的环境下，在 Project__Setting__Link 中添加 SHARED_VAR.lib 和 PROCESS_DLL.lib, 各个应用程序的可执行文件，各个动态链接库的.dll 文件都应该存在一个文件夹中。

3.6 多任务是其他语言的应用程序

大部分的多任务是 VC++的应用程序，甚至于 VB 的应用程序也可以通过动态链接库来访问(使用和修改)VC 应用程序中的共享数据。在 VB 的应用程序中需要以下的说明：

```
Private Declare Function GetVarValue Lib "SHARED_VAR" (ByVal astr As String, ByVal aint As Integer, ByVal adub As Double) As Double
```

在动态链接库中需要定义 GetVarValue(char * int,double)函数，以便 VB 的应用程序可以通过变量名字来取得共享变量的值(第二个参数为 1)；也可以修改共享变量的值(第二个参数为 2)。并且进行 extern "C" __declspec(dllexport) 和 _stdcall 的说明；格式为：

```
extern "C" __declspec(dllexport)
double _stdcall  GetVarValue(char * ,int ,double )；
```

3.7 内存共享的另一个问题：同步

多个进程可以拥有同一信号灯、事件或互斥等对象的句柄，所以这些对象可用于实现进程之间的同步。互斥方法是其中较为简单的一种可直接利用 VC++提供的互斥对象实现。

```
CMutex mMutex1(TRUE, "Info1_Mutex"); // 互斥对象
CSingleLock mSLock1 ( & mMutex1 );
```

子线程的工作:

```
mSLock1.Lock( );
……
    mSLock1.Unlock( );
```

4 结 束 语

随着硬件速度的发展和软件规模的扩大，多任务操作系统下进程之间的通信量也在增加，进程之间通信的手段多种多样，其中内存共享方法，再通过动态链接库共享数据，是一种简便、实用的共享数据的方式，较好地实现了实时性;并且也能使不同语言的应用程序之间共享数据。不仅可以在 WindowsNT 系统下实时动态模拟环境实现多任务间的数据共享，也可以用于其他的情况下，如 Windows98、Windows2000 等环境，而且具有实时、快速和高效的特点。

参 考 文 献

1 朱友芹.新编 Windows API 参考大全.北京:电子工业出版社,2000

2 刘文智.Visual C++ 6.0 教程.北京:电子工业出版社, 2000

3 扬锡林.Visual Basic 编程高手.北京:北京大学出版社, 2000

4 Chen Wenyu, Wei Chunmin. Inter-control of multitasking on dynamic simulation environment for Windows NT. Journal of University of Electronic Science and Technology of China, 2001,30(5): 507-510[陈文宇,卫春敏. WindowsNT 系统下实时动态模拟环境中多任务的控制.电子科技大学学报,2001,30(5):507~510]

5 Jiang Xuping, Yao Aiqun. Dynamic data exchange technique and its application. Journal of University of Electronic Science and Technology of China, 1997,26(supp): 329-334[姜旭平,姚爱群.Windows 下的应用程序动态数据交换过程.电子科技大学学报,1997,26(增):329~334]

The Data Sharing of Multitasking in the Real-time Dynamic Simulation Environment

Chen Wenyu

(College of Computer Science and Engineering, UEST, of China Chengdu 610054)

He Yuanqing

(Dept. of Basic Studies, CAFF of Chian Guanghan 618300)

Abstract Owing to the demand of the data sharing between different application programs, based on the feature of Windows NT and VC++, a method for data sharing by using DLL is proposed. The method solves the problem of data sharing of multitasking on real-time dynamic simulation environment for Windows NT. The shared data are defined in one DLL file and which can be shared by every application program, even the application program with different languages. The application program can also access the shared data indirectly. The method is convenient and useful.

Key words shared data; DLL; multitasking; mode of access variable