基于 VC++的进程通信技术研究*)

徐江峰[1] 张战辉[1] 杨 有[2]

(郑州大学信息工程学院 郑州 450001)[1] (重庆师范大学数学与计算机科学学院 重庆 400047)[2]

摘 要 本文对多种进程间通信技术进行了研究,给出了 VC++中实现进程通信的相关函数,分析和对比了各种 IPC 技术的特性,并通过实例说明了基于 VC++的无连接套接字进程通信机制的实现方法。

关键词 进程,进程通信,套接字

Research of Inter-Process Communication Technology Based on VC++

XU Jiang-Feng[1] ZHANG Zhan-Hui[1] YANG You[2]

(School of Information and Engineering, Zhengzhou University, Zhengzhou 450001)[1]

(College of Mathematics and Computer Science, Chongqing Normal University, Chongqing 400047)[2]

Abstract In this paper many kinds of IPC technologies are discussed, and some VC functions referenced are introduced. Comparisons and analyses about these technologies are done. An IPC implementation of unconnective socket is given at last.

Keywords Process, Process communication, Windows socket

1 引言

随着计算机技术的不断发展,计算机系统中运行的应用程序功能日益强大,孤立存在的、独立运行的应用程序越来越少,越来越多的程序相互间需要共享数据和不断的交换信息。即使在同一应用程序中,也常常包含多个进程,它们之间同样需要相互通信、共享和交换数据,共同完成程序功能。Windows 中一个进程的地址空间内的数据是私有的,别的进程不可见。不同类型的进程通信手段也常常是不同的,在程序设计中采用的进程通信技术不同,对应用程序的功能实现和运行效率都有着重要影响。

目前进程间通信(IPC: Inter-Process Communication)技术主要有 WM_COPYDATA 消息、剪贴板(Clipboard)、文件映射(File Mapping)、动态链接库(DLL)、管道(Pipe)、邮槽(Mail Slot)、动态数据交换(DDE)、远程过程调用(RPC)、NetBIOS、Windows Sockets、COM/DCOM 等。本文将对这些技术进行分析和研究,给出 VC++的相关函数,并通过实例说明其实现方法。

2 IPC 技术及 VC 相关函数

2.1 WM_COPYDATA 消息

在 Windows 平台下实现 IPC 最简单的方式就是消息。当一个进程向另一个进程发送数据时,发送方需要发送 WM_COPYDATA 消息给接收方,并提供接收方窗口句柄和含有所传数据指针的数据结构的地址。

在 WM_COPYDATA 消息的发送过程中使用了文件映射:首先把发送进程地址空间的数据复制到文件映射对象,然后在接收进程中打开该文件映射对象的一个视图。

VC 相关函数:消息发送函数 SendMessage(不可以用 PostMessage)、结构体 COPYDATASTRUCT。

2.2 剪贴板(Clipboard)

剪贴板是 Windows 应用程序间进行通信的常见方式之一,也是最容易实现的方式。它使用了全局内存在进程间传递数据,是一种较松散的数据交换机制,系统中支持剪贴板数据格式的所有应用程序都共享剪贴板。剪贴板操作一般是用户驱动的,也就是说应用程序只有在得到用户的指令后才可以把数据发送到剪贴板或从剪贴板接收数据。所以剪贴板常用于有用户参与的交互式应用。

VC 相关函数:OpenClipboard、GetOpenClipboardWindow、CloseClipboard、EmptyClipboard、GetClipboardOwner、SetClipboardData、GetClipboardData 等。

2.3 文件映射(File Mapping)

文件映射能使进程把一个文件或部分文件内容当作进程地址空间中的一块内存来对待。因此,各个进程在自己的地址空间里接收内存指针。由于 Win32 API 允许多个进程访问同一文件映射对象,因而通过这些指针不同进程就可以读取或修改文件的内容,实现进程间的通信,并且文件映射允许进程高效率地处理大文件。

VC 相关函数:创建文件映射对象 CreateFileMapping、将部分或整个文件映射到进程的地址空间 MapViewOfFileEx、当使用完文件映射对象时销毁所有文件视图 UnmapViewOfFile。

2.4 动态链接库 DLL (Dynamic Link Library)

动态链接库(DLL)可以包含函数代码、数据以及 Windows 资源,它简化了数据和资源的共享,多个应用程序可以同时访问内存中的一个 DLL 拷贝。所以 DLL 可以通过自己的全局数据与多个调用它的应用程序共享数据,从而达到进程间通信的目的。

*)本课题得到河南省教委自然科学基金支持(2006520014)。**徐江峰** 博士,副教授,主要研究方向为数据加密技术。**张战辉** 硕士,研究方向为计算机应用技术。

由于Win32API的多任务特点，DLL需要使用信号量或其它同步对象来控制对共享内存的访问。尽管DLL可以进行全局数据的共享，对于共享内存来说文件映射比较好，因为文件映射效率高并且提供了对共享数据的访问控制，例如客户机可以被限制只能读文件映射对象。

VC相关代码：例如想创建一个字符数组，这个数组被加载该DLL的所有进程共享。编码如下：

```
#pragma data_scg("common")
Char strSharedArray 1024 "zhang";
#pragma data_scg()
```

必须先对该数组初始化，否则编译器将把它放在普通未初始化的段中而不是放在"common"中[1]。不同进程可以通过对字符数组strSharedArray的操作来实现IPC。

2.5 管道(Pipe)

管道可以看成是一种有限大小的特殊文件，以先进先出形式进行读写。从用户的角度来看，数据从管道的一端写入，从另一端读出。系统为管道的读写提供同步机制。尽管系统确保处于管道两端的进程处于连接状态，但避免进程死锁仍是应用程序的责任[2]。

管道分两种：匿名管道(Anonymous Pipe)和命名管道(Named Pipe)。匿名管道只能用于父子进程或兄弟进程之间通信，并且它一般被用来重定向子进程的标准输入或输出，使它可以与父进程交换数据。而命名管道可用于同一系统或处于网络上的不同系统之间的通信。

VC相关函数：创建匿名管道CreatePipe、创建命名管道CreateNamedPipe、连接命名管道ConnectNamedPipe、等待连接WaitNamedPipe等。

2.6 邮槽(Mail Slot)

邮槽(Mail Slot)是基于广播通信体系设计出来的，它采用的是无连接的、不可靠的数据报(如TCP/IP协议中的UDP报文)传输。邮槽是一种单向通信机制，创建邮槽的服务器进程读取数据，打开邮槽的客户机进程写入数据。写入的数据一直放在邮槽中直到服务器进程读取为止。一个进程既可以是邮槽服务器也可以是邮槽客户机，因此可以建立两个邮槽实现进程间的双向通信。

邮槽客户机可以给本地计算机上的邮槽、其它计算机上的邮槽或指定网络区域中计算机上有同样名字的邮槽发送数据，即邮槽可以是一对多的，与广播类似。为保证邮槽在各种Windows平台下都能够正常工作，应将所发送数据的长度限制在424字节以下。

VC相关函数：创建邮槽CreateMailslot、查看油槽GetMailslotInfo、设置油槽SetMailslotInfo等。

2.7 动态数据交换DDE (Dynamic Data Exchange)

DDE是一组消息和规则的集合，它在进程间使用共享内存交换数据，可以用来进行一次性的数据传输，也可以进行连续的数据更新。动态数据交换(DDE)机制克服了利用剪贴板进行数据通信时数据静态的缺点，实现了进程间数据的动态传递，无须用户介入[3]。

Win32API也提供了动态数据交换管理库DDEML。基于Win32的应用可以用DDEML来共享数据。DDEML通过提供消息和函数简化了给Win32应用增加DDE支持的工作，一个应用可以使用DDEML函数来管理DDE会话，而不用直接发送和处理DDE消息。

VC相关函数：ImpersonateDdeClientWindow、FreeDDElParam、DdeSetQualityOf Service、PackDDElParam、ReuseDDElParam、UnpackDDElParam，消息有WM_DDE_ACK、WM_DDE_ADVISE、WM_DDE_DATA、WM_DDE_EXECUTE等。

2.8 远程过程调用RPC (Remote Procedure Call)

RPC集成了三种编程模型：通过编写过程和库开发C应用的模型；使用高性能计算机作为网络服务器为客户提供特定服务的模型；C/S模型，它在客户端处理UI，服务端处理数据的存储、查询和操作。

RPC机制强调通信的两个应用程序所处的环境和平台必须是相同的，而且都处于运行状态。做远程调用时，两者要先建立连接，通讯链路质量对它的效果影响很大[4]。

作为一种创建分布式应用的工具，RPC具有以下优点：它是一个广泛流行的编程模型，为开发者隐藏了网络接口的很多细节，解决了不同种类网络上的数据转换问题。

在VC中开发RPC应用非常复杂，详见MSDN | 平台SDK | Networking and Distributed Services | Remote Procedure Call(RPC) | Overviews | Tutorial。

2.9 NetBIOS(Network Basic Input/Output System)

IBM在20世纪80年代开发了NetBIOS，它包含一些面向连接的API，用来开发C/S模式应用程序。随后IBM把NetBIOS扩展为NetBEUI，微软又在Windows操作系统下把其扩展为NBF(NetBIOS Frame)。因为传输层协议和NetBIOS API接口独立，所以通过协议封装，Net2BEUI还可运行在具备路由能力的IPX和TCP/IP协议网络中，使得NetBIOS应用程序在路由网络环境中运行。NetBEUI主要为部门级的LAN进行性能优化，所以在局域网中NetBEUI速度快、占用内存较少、性能高。

一个基于Win32的应用可以通过NetBIOS接口与网络中其它计算机上的应用进行交互。但是NetBIOS接口主要是为那些使用IBM NetBIOS 3.0的应用提供的，在其它应用中可以使用邮槽、命名管道等完成类似功能，并且比NetBIOS更灵活。

VC中NetBIOS接口只提供了一个函数Netbios来解释和执行指定的网络控制块。详见MSDN | 平台SDK | Networking and Distributed Services | The NetBIOS Interface。

2.10 Windows Sockets

Windows Sockets规范为Microsoft Windows定义了一个二进制兼容网络编程接口。Windows Sockets基于Berkeley Software Distribution(BSD，4.3版)中的UNIX套接字实现。该规范包括针对Windows的BSD样式套接字例程和扩展。通过使用Windows Sockets，应用程序能够在任何符合Windows Sockets API的网络上通信。

许多网络软件支持网络协议下的Windows Sockets，这些协议包括：传输控制协议/网际协议(TCP/IP)、Xerox网络系统(XNS)、Digital Equipment Corporation的DECNet协议和Novell Corporation的互联网包交换协议/顺序分组报文交换协议(IPX/SPX)等。虽然目前的Windows Sockets规范定义了TCP/IP的套接字抽象化，但任何网络协议都可以通过提供自己版本的、实现Windows Sockets的动态链接库(DLL)来满足Windows Sockets。

VC相关函数：加载套接字库WSAStartup、卸载套接字库WSACleanup、创建套接字socket、绑定套接字bind、监听套接字listen、允许连接accept、发送数据send和sendto、接收数据recv和recvfrom、关闭套接字closesocket、连接服务端

connect 等等。

2.11 COM/DCOM(Distributed Component Object Model)

COM(组件对象模型)是 OLE 对象进行相互交互的基础,它是微软的第三代组件结构。其第一代组件 DLL 和第二代组件 Windows 开放系统体系结构(WOSA)提供的均是 C 语言函数调用接口,而 COM 基于对象技术并扩展了对象的功能。

COM 是实现一个软件组件按照某种约定和规则与另一软件组件交互作用的机制。COM 采用二进制标准,不仅使不同程序设计语言实现的 COM 程序能交互操作,而且一种程序设计语言创建的组件可被另一程序设计语言使用。

DCOM(分布式组件对象模型)把 COM 技术扩展到网络,通过使用 RPC 处理低层通信。DCOM 比目前的 OLE 特性增加了安全性和可升级性,而这两个特性是开发大型分布式组件软件所必需的。

MFC 提供了很多 OLE 类,例如 OLE 容器类 COleDocument,OLE 服务类 COleServerDoc,OLE 公用对话框类 COleDialog,OLE 控制类 COleControlModule,OLE 拖放和数据传送类有 COleDropSource,这些类大大简化了基于 VC 的 COM 编程。

3 IPC 技术特性比较

通过上述分析,可以得到各种 IPC 技术所具有的特性,表 1 对它们具有的一些特性进行了比较。

表 1

特性 / IPC技术		网络应用	安全性	网络接口	工业标准	同步	可升级性
命名管道		支持	有	高级	否	不需要	无
DCOM		支持	有	高级	是	不需要	有
RPC		支持	有	高级	是	不需要	有
Windows Sockets	UDP 和 IPX	支持		中级	否	不需要	
	TCP 和 SPX	支持		中级	否	不需要	
邮槽		支持	有		否	不需要	无
匿名管道		不支持	有	高级	否	不需要	无
文件映射		不支持	有	不是	否	需要	无
DLL		不支持	无	不是	否	需要	无
WM_COPYDATA 消息		不支持	有	不是	否	不需要	无
剪贴板		支持	无	不是	否	用户驱动	无
DDE		不支持	有		否	不需要	无

注:部分表格未填,有待进一步研究。

从表中可以看出,文件映射和 WM_COPYDATA 消息是单机上非常有效的 IPC 技术,但前者需要同步,后者不用。文件映射对大文件非常有效,并且提供了对共享数据的访问控制。剪贴板对于要传送的数据的类型有限制,并且是用户驱动的,应用程序不能在用户不知情的情况下使用剪贴板传送数据。管道在数据量比较大的情况下比其它 IPC 技术有效,并且在单机情况下,无论传送多大数据,所用时间基本上是相同的[6]。邮槽对于进程间收发短信息比较容易,并且能在一个网络上进行数据群发。RPC 是函数级接口,它支持自动数据转换,可以与非 Windows 系统交互,但程序开发比较麻烦。Windows Sockets 是协议无关的接口,它主要利用底层协议的交互能力。DCOM 是工业标准,适合开发大型分布式组件软件。

4 基于 VC++ 的无连接套接字进程通信机制实现

下面我们给出一个基于 VC++ 的无连接套接字进程通信机制实现,以说明上述技术的应用方法。

UDP 服务端代码:

```
void main()
{
    WORD wVersionRequested = MAKEWORD( 2, 2 );;
    WSADATA wsaData;
    int err = WSAStartup( wVersionRequested, &wsaData );
    if ( err != 0 ) return;
//下面绑定一个本机端口到套接字上
     SOCKET sockSrv=socket(AF_INET,SOCK_DGRAM,0);
     SOCKADDR_IN addrSrv;
     addrSrv.sin_addr.S_un.S_addr=htonl(INADDR_ANY);
     addrSrv.sin_family=AF_INET;
     addrSrv.sin_port=htons(6000);
     bind(sockSrv,(SOCKADDR *)&addrSrv,sizeof(SOCKADDR));
//下面接收数据并输出到屏幕上
   SOCKADDR_IN addrClient;
   int len=sizeof(SOCKADDR);
   char RecvBuf[100];
   recvfrom(sockSrv,RecvBuf,100,0,(SOCKADDR *)&addrClient,
   &len);
   printf("%s\n",RecvBuf);
   closesocket(sockSrv);
   WSACleanup();
}
```

UDP 客户端代码:

```
void main()
{
    WORD wVersionRequested = MAKEWORD( 2, 2 );;
    WSADATA wsaData;
    int err = WSAStartup( wVersionRequested, &wsaData );
    if ( err != 0 ) return;
//下面绑定端口
     SOCKET sockClient=socket(AF_INET,SOCK_DGRAM,0);
     SOCKADDR_IN addrSrv;
     addrSrv.sin_addr.S_un.S_addr=inet_addr("127.0.0.1");
     addrSrv.sin_family=AF_INET;
     addrSrv.sin_port=htons(6000);
//下面发送"hi"到服务端
     sendto(sockClient,"hi",strlen("hi")+1,0,(SOCKADDR *)
     &addrSrv,sizeof(SOCKADDR));
     closesocket(sockClient);
     WSACleanup();
}
```

结束语 本文对各种进程通信机制进行了分析和研究,给出了 VC++ 的相关函数,最后给出了一个基于 VC++ 的无连接套接字进程通信机制实现。通过上述分析可以看出,虽然 IPC 技术有很多,但它们各有优缺点和应用范围,在程序开发中需要结合实际选择合适的技术。例如,是否要进行网络通信,传输的数据量是否很大,是否需要用户参与,是否需要进行一对多通信等。

参 考 文 献

1 陈雄,周文胜,陈鹤鸣. Windows 下的进程间通信[J]. 南京邮电学院学报(自然科学版),2000,20(2):51~54
2 胡小龙,江海花. Windows 操作系统进程通信机制[J]. 计算机应用研究,2002,12:119~121
3 许欢庆,赵晨. Win32 进程间通信技术[J]. 西北纺织工学院学报,2000,14(4):351~362
4 朱辉生. 进程及应用程序间通信的实现技术[J]. 计算机应用与软件,2004,21(1):118~120
5 徐晓刚,高兆法,王秀娟. Visual C++ 6.0 入门与提高[B]. 清华大学出版社,1999
6 孙叶萌. 单机内进程间三种通信方法效率的比较[J]. 长春理工大学学报,2003,26(3):81~83