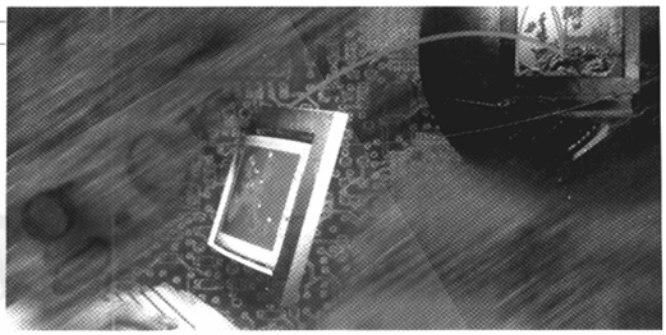
虚拟设备 VxD 异步通信技术的研究与应用

邸晓奕 (西安 61542 部队网络中心 710016)
刘雪梅 (陕西省农电管理局设计室 710077)



摘要：本文介绍了在WINDOWS多任务系统中，运行于系统核心层的虚拟设备驱动程序VxD与运行于应用层的进程之间的通信方式，并给出具体的异步通信应用实例。

关键词：VxD VMM VM VtoolsD

在Windows多任务操作系统中，计算机的物理设备可同时被多个进程使用，而这些进程运行于系统的Ring3层，它们无法直接与底层的物理设备进行信息交换。要想在Ring3层完成对系统物理层的访问，必须借助于VxD技术。本文将介绍如何通过与VxD进行异步通信，从而使运行在ring3层的应用程序具有访问硬件的能力，并以VxD异步通信为例，给出一个具体的应用实例。

1 Windows的虚拟环境及VxD技术概述

Windows的系统核心由虚拟设备驱动程序VxD，虚拟机管理器VMM(Virtual Machine Manager)和虚拟机VM(Virtual Machine)三部分构成，其中虚拟机VM分为DOS虚拟机和系统虚拟机两种。每个虚拟机都有独立的地址空间、I/O空间、寄存器、堆栈、局部描述符、中断向量表和执行优先权。每个DOS应用程序在独自的DOS虚拟机中运行，而Windows应用程序均在系统虚拟机中运行。虚拟机管理器VMM包含了所有基本的系统功能，如任务调度、虚拟内存操作、程序装入及终止、进程间通信，此外，还负责处理主要的中断处理及例外情况。虚拟设备驱动程序VxD最初是用来支持Windows下硬件设备的管理，扩展系统功能的一类程序，它以DLL的形式链接入操作系统的核心层Ring0，运行在Ring3层的应用程序通过与Ring0层的VxD通信获得访问硬件的能力。所有VxD的运行都在VMM的监控下。VxD、VMM和VM的关系如图1所示：

设备驱动程序VxD是管理系统软硬资源的二进制可执行代码。它是1个32位保护模式下的可执行DLL，在16位的Windows 3.x中，VxD一般具有后缀名.386，并且在SYSTEM.INT文件的[386Enh]节中静态地加载。在Windows 98中，由于即插即用的引入，VxD的功能也被增强了。现在VxD可以动态地加载或卸载，以节省系统资源。在很大程度上VxD起到了原来BIOS的作用。

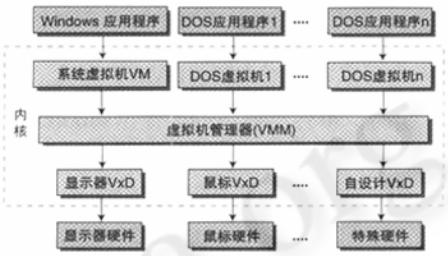


图 1 VxD、VMM和VM的关系

2 虚拟设备驱动程序VxD的开发过程

过去的VxD全部用Intel汇编语言编写，编程者需要有很好的x86汇编语言功底，还得对Windows体系结构有相当的了解。如今由美国Vireo Software公司推出的VtoolsD for Windows 95开发工具包提供了对VxD编程的全线C/C++类库支持，绝大多数的VMM和VxD的服务都可以通过类库中的成员函数来实现。

用VtoolsD工具包中的QuickVxD可为编程者生成VXD的C/C++程序框架（生成*.h，*.cpp，和*.mak三个文件），用户在此基础上添加必要的代码，最后再用VC++编译器经编译、链接生成相应的设备驱动程序VxD。

创建虚拟设备驱动程序的基本步骤如下：

2.1 利用VtoolsD工具包中的QuickVxD生成VxD框架

在QuickVxD可视化环境中，程序员只需在VxD快速生成对话框中选定和定义所需VxD的基本特征，如实现语言(C或C++)、设备参数、控制消息、调用方式、VxD服务以及其他一些文件名和类名后，即可自动生成指定语言的VxD源程序框架。如果要求VxD能够被动态加载并且要与Ring3层进行通信，则必需在VxD控制消息框中选中SYS_DYNAMIC_DEVICE_EXIT、SYS_DYNAMIC_DEVICE_INIT、W32_DEVICEIOCONTROL三项。

2.2 重载类成员函数

自动生成的VxD源程序中包含三个非常主要的类：VDevice、VvirtualMachine、VThread。其中VDevice用于响应VMM或由其他VxD传来的消息，VVirtualMachine用于响应VM的控制消息，VThread用于对特定线程的创建、终止进行控制。当Windows系统装载VxD时，Vdevice类的一个实例将自动被创建，而VvirtualMachine和Vthread类的实例必需由手工创建。QuickVxD为这些类的成员函数提供缺省的消息处理方式，程序员只需重载感兴趣的成员函数即可处理相应的控制消息。

2.3 添加必要的代码

利用VxD可以实现对中断、DMA、热键、I/O端口、内存分配、定时、事件等服务的处理。VtoolD为每项服务提供相应的类，所有这些服务处理类都必须从VDevice类中派生。程序员根据具体应用的需要创建所需类的实例，重载缺省的消息处理方式，添加必要的成员函数。

2.4 生成VxD程序

虚拟设备驱动程序可以在MS VC++集成环境中编译生成VxD文件。在VC++集成环境中选择Project＼Setting菜单选项，在工程设置对话框的Build command line选项中设置"NMAKE /f FileName.mak"，在Output file name选项中设置"FileName.vxd"。设置完毕后，进行编译连接即可生成VxD程序。

2.5 编写Ring3层应用程序

生成的VxD程序需要Ring3层的应用程序来加载和测试。对于可动态加载的VxD，应用程序通过CreateFile()函数动态加载VxD，用CloseHandle()函数动态卸载VxD，用DeviceIoControl()函数与VxD通信。

3 VXD通信技术的实现

作为设备驱动程序，VxD运行于Ring0层，所以没有它不能完成的事情，Ring3层应用程序利用VxD底层硬件访问能力，通过与VxD通信来完成自己不能完成的任务。与VxD通信主要接口函数有：DeviceIoControl，CreateEvent，GetOverlappedResult，OnW32DeviceIoControl，VWIN32_DIOCCompletionRoutine(具体的函数定义请查阅有关资料)。

(1) Ring3层的应用程序通过DeviceIoControl函数使VMM发送W32_DEVICEIOCONTROL消息给VxD。该函数的定义如下：

BOOL DeviceIoControl(

HANDLE hDevice, 用CreateFile函数加载的VxD设备句柄

DWORD dwIoControlCode, Ring3层程序向VxD传递的命令参数

LPVOID lpInBuffer, Ring3层程序向VxD传递的数据输入缓冲地址

DWORD nInBufferSize, Ring3层程序向VxD传递的数据输入缓冲字节数

LPVOID lpOutBuffer, VxD返回数据的缓冲地址

DWORD nOutBufferSize, VxD返回数据的缓冲区字节数

LPDWORD lpBytesReturned, VxD实际返回数据的字节数

LPOVERLAPPED lpOverlapped 用于异步通信的OVERLAPPED结构地址

)

应用程序通过dwIoControlCode参数向VxD发送控制命令，通过lpInBuffer缓冲区向VxD发送数据，通过lpOutBuffer缓冲区从VxD接收数据。

(2) 应用程序通过CreateEvent函数创建异步通信事件，该异步事件通过OVERLAPPED结构的hEvent传递给VxD。

(3) 在异步通信方式中，应用程序向VxD发送控制信息后，通过调用GetOverlappedResult函数进入异步等待状态。

(4) 在VxD内部，设备驱动程序通过OnW32DeviceIoControl函数处理WIN32_DEVICEIOCONTROL消息。当应用程序加载VxD、卸载VxD以及调用DeviceIoControl时，OnW32DeviceIoControl函数将自动被触发。

(5) 当VxD完成Ring3层应用程序发出的控制命令后，通过VWIN32_DIOCCompletionRoutine函数将异步事件对象置为有信号状态，以便唤醒处于等待状态的Ring3层应用程序。

在异步通信中，CreateFile函数的fdwAttrsAndFlags参数必须含有标志FILE_FLAG_OVERLAPPED以表明系统要以异步方式访问虚拟设备。为了安全起见，VxD可用VMM服务_LinPageLock,按页锁定OnW32DeviceIoControl函数参数PIOCTLPARAMS结构中dioc_InBuf指向的应用程序输入缓冲区，dioc_OutBuf指向的输出缓冲区，lpoOverlapped指向的OVERLAPPED结构.要交换数据时，可置数据及数据总量到dioc_OutBuf及lpoOverlapped->O_InternalHigh。具体的通信流程如图2所示：

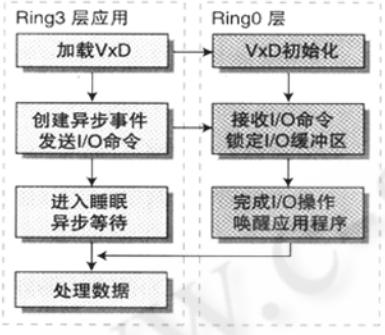


图 2 异步通信流程

4 VxD异步通信具体应用

作者在利用C30语音采集器开发语音处理软件时，需要编写VxD将语音采集器采集的语音数据提交给Windows应用程序进行处理。由于异步通信方式能较好的实现多任务处理，提高系统的整体效率，所以VxD与应用程序间采取异步通信方式。现将在语音采集系统中的VxD通信的关键代码摘录如下：

```
HANDLE hEvent, hVxd;
DWORD cbRet = 0;
OVERLAPPED ovlp = {0,0,0,0,0};
char InBuffer [255],OutBuffer [255];
int result,BufferSize = 255;
    hVxd=CreateFile("\\\\.\\My_C30vxd.vxd",0,
0,0,CREATE_NEW,FILE_FLAG_DELETE_ON_CLOSEI-
FILE_FLAG_OVERLAPPED,0);
    hEvent = CreateEvent(0, true,false, NULL);
    ovlp.hEvent=hEvent;
   DeviceIoControl(hVxd,START_COLLECT,InBuffer,
BufferSize, OutBuffer, Buffersize, cbRet, &ovlp); // 以
START_COLLECT 为标记向 V × D 发控制信号
    GetOverlappedResult(hVxd, &ovlp, &cbRet, true);// 异
步等待 VxD 操作结束
    …… // 处理 OutBuffer 中采集来的数据
    CloseHandle(hVxd);
    VxD 对控制命令的响应代码:
DWORD My_C30Device::OnW32DeviceIoControl
(PIOCTLPARAMS pDIOCParams)
{
    OVERLAPPED* pOverlap;
    switch(pDIOCParams->dioc_IOCtlCode)
    {
case DIOC_OPEN:
    ……// 语音采集器初始化处理
    break;
case DIOC_CLOSEHANDLE:
    ……// 语音采集器结束处理
    break;
   case START_COLLECT:   // START_COLLECT控制
命令的处理
      ……// 语音数据的采集
    // 向 Ring3 层应用传送语音数据，ptSoundData 为语
音数据在采集器中地址指针
      memcpy(pDIOCParams->dioc_OutBuf, ptSoundData,
255);
      (DWORD *)pDIOCParams->dioc_bytesret=255;//提
交的字节数
       pOverlap=(OVERLAPPED *)pDIOCParams->
dioc_ovrlp;
       VWIN32_DIOCCompletionRoutine(pOverlap->
O_Internal);// 置异步事件为有信号  break;
    case 其他控制命令:
      ……// 相应命令的处理
      break; }
    return 0;
    }
```

为简单起见，以上代码中所有函数的调用均未对返回值进行判断，这在实际编程中切不可如此。文章仅列举了VxD的通信部分代码，有关VxD初始化控制以及各种类的使用与相互调用关系请参阅相关资料。■