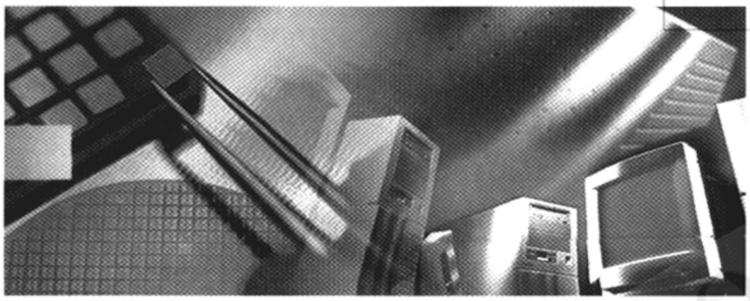
Win9x 下的 VxD 技术与应用

李湘江 （长沙交通学院信息与计算科学系 410076）



摘要：本文深入分析了 Windows 9x 的 VxD 技术的调用机制及其工作原理，介绍了VxD的特点及常用开发工具。最后探讨了 VxD 技术的应用。

关键词：VxD 技术分析 调用机制

1 VxD 结构原理

1.1 VxD 的文件结构

VxD的结构是包含下面5个段的一些组合:

(1) VxD_CODE 段：保护模式代码段(必须)。该段包含 VxD 系统控制过程、回调过程、服务和VxD的API接口函数。该段用宏 VxD_CODE_SEG 和 VxD_CODE_ENDS 定义开始和结束，也可命名为 _LTEXT。

(2) VxD_DATA 段：保护模式数据段（必须）。该段包括设备描述 DDB、服务表和部分 VxD 所用的全局数据。该段用宏 VxD_DATA_SEG 和 VxD_DATA_ENDS 定义开始和结束，也可命名为 _LDATA。

(3) VxD_ICODE段：保护模式下的初始化代码段(可选)。该段一般包括只在 VxD 初始化过程中使用的过程和服务，VMM 在初始化结束后即 Init_Complete 消息发生后丢弃此段。该段用宏 VxD_ICODE_SEG 和 VxD_ICODE_ENDS 定义开始和结束，也可命名为 _ITEXT。

(4) VxD_IDATA段：保护模式初始化数据段(可选)。该段一般包括初始化过程和服务使用的数据，VMM 在 Init_Complete 消息发生后丢弃此段。该段用宏 VxD_IDATA_SEG 和 VxD_IDATA_ENDS 定义开始和结束，也可命名为 _IDATA。

(5) VxD_REAL_INIT段：实模式初始化段(可选)。该段包含实模式初始化过程和数据，VMM 在加载 VxD 其他部分之前调用此过程，过程返回后丢弃此段，该段用宏 VxD_REAL_INIT_SEG 和 VxD_REAL_INIT_ENDS定义开始和结束，也可命名为 _RTEXT。

除实模式初始化段以外，所有代码和数据段均为 32 位平坦内存模式的保护模式段，这就是说定义在保护模式段中的过程和数据均为 32 位的偏移量。当 VMM 装载 VxD 时，按照 VxD 在内存中的实际位置修正所有的偏移量。因此，在保护模式段中使用普通 OFFSET 指令处应该使用 OFFSET32 宏，OFFSET32 宏定义的偏移量为连接器确定了正确的偏移量修正信息。

1.2 VxD 的数据结构

VMM是通过VxD的设备描述符块DDB(Device Descriptor Block)来识别的。DDB 向 VMM 提供了 VxD 的主入口点，还向应用程序和其他的 VxD 提供了入口点。VMM 利用这个主入口点将VM及Windows自身的状态通知给 VxD，然后 VxD 通过相应的工作来响应这些事件。

由于 VxD 不仅仅服务于一个物理设备或仅与一个 VM 发生联系，所以 VxD 需要产生自己支持的数据结构来保存每一个设备、每一个VM的配置和状态信息。VxD用一个或多个设备上下文结构来保存设备信息，如 I/O 端口基地址、中断向量等，VxD 将自己的每个 VM 的状态信息保存在 VMM 的 VM 控制块中。

2 VxD 调用机制

2.1 VMM 及其提供的扩展服务

VMM（Vircual Machine Manager）即虚拟机器管理程序，是在系统级核心运行的32位保护模式操作系统，是操作系统的核心。VMM 负责为计算机中运行的所有应用程序和系统进程分配所需资源，主要功能是创建、运行、监控和终止虚拟机。如它提供低级内在管理和调度服务；同时还有对虚拟设备驱动程序的服务。VxD是用来扩展 Windwos 操作系统功能的一类程序。由于VxD运行在系统的Ring 0级，拥有与操作系统同等的级别，所以我们可利用它来支持硬件设备的管

理。虚拟可编程中断控制器（VPCD）是负责管理所有硬件中断事件的程序，它本身也是一种VxD，能提供缺省的中断处理函数或者允许其他VxD重载中断处理函数。

在系统初始化过程中，一旦安装了VMM、它就会永久驻留。Windows在基本操作系统中包括700多个服务，其中接近一半服务是由VMM提供的。正常情况下，VMM提供的服务包括事件服务、内存管理服务、兼容执行和保护模式执行的服务、登录表服务、调度程序服务、同步服务、调试服务、I/O捕获服务、处理错误和中断服务、VM中断和回调服务、配置管理程序服务以及其他杂项服务。而且VMM服务的范围覆盖了最低级的操作系统需要。

每个VxD都定义一个服务表，用于识别该VxD内部函数的入口点，这些函数向别的VxD或应用程序提供服务，VxD的处理对象是VMM发来的控制信息，所以VMM是VxD或应用程序调用VxD时发挥重要作用的核心组件。Win9x系统下有众多的VxD，每个VxD可提供4种服务，即PM(保护模式)API、V86(虚拟86)API、Win32服务和VxD服务，前3种分别供应用程序在16位保护模式、V86模式以及32位保护模式下调用，VxD服务则只供其他VxD使用，用户开发的VxD可提供任意上述服务。除此之外，应用程序还可通过调用API函数DeviceIoControl与支持IOCTL接口的VxD进行通信，执行Win32 API不支持的系统低级操作。

2.2 VxD的消息处理

VxD的操作基于寄存器，所以一般用汇编语言编写，它的关键部分是一个和普通窗口的消息处理过程WndProc相类似的控制过程，不同之处在于它的处理对象是系统发来的控制消息。这些消息共51种，在VxD自加载至卸出整个生命周期内，操作系统不断向它发送各种控制消息，VxD根据自己的需要选择处理，其余的忽略。系统向VxD发送控制消息时将消息代号放在EAX寄存器中，并在EBX寄存器中放系统虚拟机(VM)句柄。

对动态VxD来说，最重要的消息有三个:SYS_DYNAMIC_DEVICE_INIT、SYS_DYNAMIC_DEVICE_EXIT以及W32_DEVICEIOCONTROL，消息代号分别是1Bh、1Ch、23h。当VxD被动态加载至内存时，系统向其发送SYS_DYNAMIC_DEVICE_INIT消息，VxD应在此时完成初始化设置并建立必要的数据结构；当VxD将被卸出内存时，系统向其发送SYS_DYNAMIC_DEVICE_EXIT消息，VxD在收到后应清除所作设置并释放相关数据结构；当应用程序调用API函数DeviceIoControl与VxD进行通信时，系统向VxD发送W32_DEVICEIOCONTROL消息，它是应用程序和VxD联系的重要手段，此时ESI寄存器指向一个DIOCParams结构，VxD从输入缓冲区获取应用程序传来的数据，相应处理后将结果放在输出缓冲区，回送给应用程序，达到相互传递数据的目的。

2.3 VxD的运行机制

但有人将VxD理解为虚拟任何驱动程序。实际上，VxD并非仅指那些虚拟化的某一具体硬件的设备驱动程序。比如某些VxD能够虚拟化设备，而某些VxD作为设备驱动程序却并不虚拟化设备，还有些VxD与设备并没有什么关系，它仅向其他的VxD或是应用程序提供服务。

VxD可以随VMM一起静态加载，也可以根据需要动态加载或卸载。正是由于VxD与VMM之间的紧密协作，才使得VxD具有了应用程序所不具备的能力，诸如可以不受限制地访问硬件设备、任意查看操作系统数据结构（如描述符表、页表等）、访问任何内存区域、捕获软件中断、捕获I/O端口操作和内存访问等，甚至还可以截取硬件中断。

在静态加载VxD的整个运行过程中，VMM控制着VxD的加载运行。

在整个初始化过程中，VxD都可以通过设置返回值阻止VxD的进一步加载。一些VMM的服务，比如VMM的初始化信息服务就只能在初始化过程中使用。

所有的静态VxD都有一个实模式初始化段，这是静态VxD开始运行的起点。当在Windows系统切换到保护模式之前VMM加载这些VxD时，就调用VxD的实模式初始化段进行初始化。通常这个实模式初始化过程可以使用ROM BIOS、MS-DOS提供的系统函数以及VMM队举供的初始化服务函数等来检查Windows环境，包括注册表和初始化文件(SYSTEM.INI)的相关设置，从而决定该VxD是否应该被加载，同时它还能为设备保留独占使用的页面和实例化数据项c当系统切换到保护模式下后就是对3个初始化消息的处理，这个在前面讲VxD的消息处理时已经讲到了。当VxD初始化完成后，它就通过消息与系统进行交互运行。

VxD动态加载时，应用程序向VxD发出DeviceIoControl调用时，DeviceIoControl的第2个参数用于指定进行何种控制，控制过程从DIOCParams结构＋0Ch处取得此控制码再进行相应处理控制码的代号和

含义由应用程序和VxD自行约定，系统预定义了DIOC_GETVERSION(0)和DIOC_CLOSEHANDLE(-1)两个控制码，当应用程序调用API函数CreateFile ("\\.\VxDName",...)动态加载VxD时，系统首先向该VxD的控制过程发送SYS_DYNAMIC_DEVICE_INIT控制消息，若VxD返回成功，系统将再次向VxD发送带有控制码DIOC_OPEN(即DIOC_GETVERSION，值为0)的W32_DEVICEIOCONTROL消息以决定此VxD是否能够支持设备IOCTL接口，VxD必须清零EAX寄存器以表明支持IOCTL接口，这时CreateFile将返回一个设备句柄hDevice，通过它应用程序才能使用DeviceIoControl函数对VxD进行控制。同一个VxD可用CreateFile打开多次，每次打开时都会返回此VxD的一个唯一句柄，但是系统内存中只保留一份VxD，系统为每个VxD维护一个引用计数，每打开一次计数值加1。当应用程序调用API函数CloseHandle(hDevice)关闭VxD句柄时，VxD将收到系统发来的带控制码DIOC_CLOSEHANDLE的W32_DEVICEIOCONTROL消息，同时该VxD的引用计数减1，当最终引用计数为0时，系统向VxD发送控制消息SYS_DYNAMIC_DEVICE_EXIT，然后将其从内存中清除。在极少数情况下应用程序也可调用API函数DeleteFile("\\.\VxDName")忽略引用计数的值直接将VxD卸出内存，这将可能给使用同一VxD的其他应用程序造成毁灭性影响，应避免使用。

3 开发工具VToolsD简介

微软为驱动程序的开发提供了设备驱动程序工具箱(DDK)，基于汇编语言的编程方式和许多VMM服务都使用寄存器的调用方式，使用起来不是很方便。没有深厚的汇编语言和硬件基础很难在短时间里开发出VxD。

VToolsD是专门用于开发VxD程序的一种工具软件，它包括VxD框架代码生成器QuickVxD、C运行库、VMM/VxD服务库、VxD的C++类库、VxDLoad和VxDView等实用工具以及大量的C、C++例程。由VC++、BC++的32位编译器编译生成的VxD程序可以脱离VToolsD环境运行。

利用QuickVxD可以方便、快捷地生成VxD的框架，即生成后缀名为h、cpp和mak的三个文件。源文件包含了运行VxD的基本组件，其中包含控制消息处理、API入口点、以及VxD服务等函数框架，并且还定义了标志，设置了编译参数，声明了类，然后在C++环境下，向生成的各个处理函数体内添加自己的代码，最后使用编译器NMAKE生成标准的VxD程序。

目前VToolsD的最新版本为3.0，它支持设备访问体系结构DAA (Device Access Architecture)，所编写的程序代码将可以在所有Windows平台（包括Win 95、Win 98以及Windows NT）上共享。

4 VxD技术的应用

4.1 VxD的实用用途

由于VxD可以虚拟根本不存在的硬件，因此可以利用VxD虚拟硬件狗来破解一些软件的加密保护；VxD工作在操作系统的最底层，所以掌握它能使开发者具备扩展操作系统的能力，比如在Win9x中截取按下的Ctrl + Alt + Delete组合键，而后弹出的是自行设计的消息对话框，实现特殊的系统控制。

实时工业控制软件的开发，以前大都是基于DOS操作系统，但是现在随着Win9x的普及，需要开发基于Win9x的工控软件，利用VxD的功能就可实现Windows下的实时工控。在很多情况下，这可以获得比较满意的实时效果。

通过VxD作中介，可使DOS TSR、Win16应用程序、Win32应用程序之间产生协作，可打破Win32应用程序4GB独立线性地址空间带来的限制等。

4.2 基于VxD的实时反病毒技术

目前国内的Win9x平台反病毒产品大多属静态反病毒软件，指导思想是“以杀为主”，这一方式的缺点是病毒在被清除之前可能早已造成了严重危害。一个好的反病毒软件应该是“以防为主，以杀为辅”，在病毒入侵时就把它清除掉，这就是实时反病毒技术。

Win9x使用的是Ring 0和3两个保护级。系统进程运行于Ring 0，因而具有对系统全部资源的访问权和管理权；而普通用户进程运行于Ring 3，只能访问自己的程序空间，不允许对系统资源进行直接访问许多操作受到限制。显然这种普通用户进程是无法胜任实时反病毒工作的，必须使后台监视进程运行在Ring 0优先级，实现这一目的基础就是VxD技术。应用程序通过使用动态加载的VxD，间接获得了对Win9x系统的控制权，但要实现对系统中所有文件I/O操作的实时监视，还要用到另一种关键技术-FileHooking，通过挂接一个处理函数，截获所有与文件I/O操作有关的系统调用。Win9x使用32位保护模

（下转第20页）

(上接第29页)

式可安装文件系统(IFS)，由可安装文件系统管理器(IFSManager)协调对文件系统和设备的访问，它接收以Win32 API函数调用形式向系统发出的文件I/O请求，再将请求转给文件系统驱动程序FSD，由它调用低级别的IOS系统实现最终访问。每个文件I/O API调用都有一个特定的FSD函数与之对应，IFSManager负责完成由API到FSD的参数装配工作，在完成文件I/O API函数参数的装配之后转相应FSD执行之前，它会调用一个称为FileSystemApiHookFunction的Hooker函数。通过安装自己的Hooker函数，就可以截获系统内所有对文件I/O的API调用，并适时对相关文件进行病毒检查，从而实现实时监控。■