Windows环境下的设备驱动程序设计

张惠娟　周利华　翟鸿鸣　编著

西安电子科技大学出版社

2002

内 容 简 介

本书讲述了 Windows 系统下设备驱动程序的开发方法。全书分为三部分，共 10 章。第一部分主要介绍了 VxD 驱动程序，以及利用 VToolsD 开发 VxD 程序的方法；第二部分和第三部分分别介绍了 Windows NT 和 Windows 2000 下的设备驱动程序基础知识，以及利用 DriverWorks 开发 NT 式和 WDM 驱动程序的方法。

本书系统介绍了在不同的操作系统下设备驱动程序开发的方法，内容实用，是一本很好的学习设备驱动程序设计的书籍。

本书适合于任何想学习 Windows 系统下设备驱动程序开发的人员，尤其适合于初步涉及设备驱动程序开发的人员。

图书在版编目（CIP）数据

Windows 环境下的设备驱动程序设计/张惠娟等编著.

西安：西安电子科技大学出版社，2002.5

ISBN 7-5606-1127-3

Ⅰ.W… Ⅱ.张… Ⅲ.窗口软件，Windows-驱动程序-程序设计 Ⅳ.TP316.7

中国版本图书 CIP 数据核字（2002）第 028247 号

策　　划　陈宇光

责任编辑　杨宗周

出版发行　西安电子科技大学出版社(西安市太白南路 2 号)

电　　话　(029)8227828　　邮　编　710071

http://www.xduph.com　　E-mail: xdupfxb@pub.xaonline.com

经　　销　新华书店

印　　刷　西安文化彩印厂

版　　次　2002 年 5 月第 1 版　2002 年 5 月第 1 次印刷

开　　本　787 毫米×1092 毫米　1/16　印张 22.25

字　　数　529 千字

印　　数　1～4 000 册

定　　价　28.00 元

ISBN 7-5606-1127-3/TP・0571

XDUP 1398001-1

如有印装问题可调换

本书封面贴有西安电子科技大学出版社的激光防伪标志，无标志者不得销售

前　言

Windows 采用特有的系统保护措施，使得在 Windows 操作系统环境下，设备驱动程序的开发模式和开发方法不同于在 DOS 环境下的，且开发难度较大。目前，市面上出售的 Windows 设备驱动程序开发书籍比较少，而且，大多数是译本，内容有一定深度，要求读者必须具有相当的专业知识。这种现状，从某种程度上来说，是一种断面，使得学习设备驱动程序开发的读者很是茫然，感觉缺少循序渐进的学习过程。

以上这种感觉，也是当初我在学习设备驱动程序开发时遇到的问题，在这两年的设备驱动程序开发、教学过程中，我也深深感觉到这一点。如何能以一种系统的、循序渐进的、简单的方式介绍这门深奥的知识，让学习者轻松入门，然后根据自身的需要，开发 Windows 系统的设备驱动程序，这就是我写这本书的原动力。

本书主要介绍了 Windows 9x、Windows NT、Windows 2000 操作系统环境下的设备驱动程序模型结构、开发、链接、调试方法。

本书对读者知识层次要求比较低。以前，Windows 系统下设备驱动程序的开发采用 DDK 工具，这种方法对开发人员要求比较高，从而使得许多开发人员望而却步。现在，有许多第三方公司提供的开发工具，使得设备驱动程序开发相对简单了些。本书对驱动程序开发方法的介绍，采用的是第三方公司提供的工具，要求读者具有 C 或者 C++编程经验和面向对象技术的概念，并具有一定的硬件知识。

从内容上来说，本书共分为三部分，分别系统介绍了 VxD 设备驱动程序、NT 式设备驱动程序及 WDM 设备驱动程序的模型、开发、编译链接、调试方法，内容通俗易懂。具体来说，本书内容如下：

第一部分：虚拟设备驱动程序 VxD。这部分包括四章内容，分别介绍了 Windows 9x 系统的体系结构、VxD 设备驱动程序结构、VxD 设备驱动程序开发工具 VToolsD、VxD 设备驱动程序实例。使得读者首先能较多地了解 Windows 系统的体系结构，然后了解 VxD 设备驱动程序，最后通过例子来深入理解 VxD 设备驱动程序的开发方法。

第二部分：Windows NT 设备驱动程序。在这部分中，包括三章内容，分别介绍了 Windows NT 操作系统、NT 式设备驱动程序 、NT 式设备驱动程序的开发工具 DriverWorks、常用的 NT 设备驱动程序开发方法以及 NT 设备驱动程序开发实例等。

第三部分：WDM 设备驱动程序。WDM 设备驱动程序模型是 Windows 2000 环境下的设备驱动程序模型，它和 NT 式设备驱动程序很相似，但也有很大的不同之处。这部分也包含有三方面内容，在延续第二部分 NT 设备驱动程序基础上，介绍了 WDM 驱动程序的模型特点、开发方法以及一些驱动程序例子。

在本书的编写过程中，得到了西安电子科技大学多媒体研究所同事们的很多帮助，尤其是我的导师周利华教授的支持、建议，在这里表示衷心的感谢。也非常感谢西安电子科技大学出版社的领导、编辑和有关人员的支持和帮助。

在这里，我也非常感激我的母亲，有她承担家务和对孩子的教育，才使得我能有更多的时间来做这件事情。我非常感谢我的先生，他给我提供了很多的建议和资料。

由于编者水平有限，时间也比较仓促，书中或许会有不足之处，殷切希望广大读者的指正和批评。

张惠娟

2002 年 4 月

目　录

第一部分　虚拟设备驱动程序 VxD

第二部分　Windows NT 驱动程序

第三部分　Windows 2000 驱动程序WDM

第一部分

虚拟设备驱动程序 VxD

- Windows 9x 操作系统体系结构
- Windows 9x 虚拟设备驱动程序 VxD
- VxD 驱动程序开发方法
- VxD 驱动程序实例

第一章 Windows 9x 操作系统体系结构

1.1 Intel CPU的工作模式

从 80386 开始，Intel CPU 提供了三种 CPU 工作模式：实模式、保护模式和虚拟 86 模式。

我们以前使用的 MS - DOS 操作系统工作在实模式下，目前流行的 Windows 98 等操作系统是工作在保护模式下。为了在 32 位操作系统环境下能运行 MS - DOS 程序，Windows 98 中的 DOS 程序是工作在一种特殊的保护模式下——虚拟 86 模式，从而使得 CPU 的高级性能得到更好的发挥。相对于实模式来说，保护模式程序运行更为安全。

了解 CPU 工作模式，对于理解操作系统的工作机制，理解在操作系统基础上工作的设备驱动程序很重要。下面，分别介绍这三种工作模式及其特点。

1.1.1 实模式

实模式也称实地址模式，顾名思义，在这种模式下工作的程序，使用到的逻辑地址就是实际物理地址。

在 Windows 95、Windows 98 操作系统中，开机时，CPU 处于实模式工作方式中，这时，CPU 关闭所有保护功能，地址空间被限制在 1 MB 物理内存中。

在实模式下，内存管理采用段内存技术，不采用分页管理技术。这种技术将 1 MB 的物理空间分为 16 个段，每个段为 64 KB，段地址的基地址存储在段寄存器中。实际物理地址就是段寄存器中的数值左移 4 位，再加上偏移量形成 20 位的物理空间地址，从而实现 1 MB 地址空间的访问。

1.1.2 保护模式

保护模式是 80386 中常用的工作方式，也是最为复杂的工作模式。在这种模式下，实现了多任务宏观并行工作，实现了内存的分段、分页管理。

在保护模式中实现了多任务系统，系统同时运行着多个任务或者程序。为了防止各个程序之间的相互干扰，CPU 采用了一种硬件机制，以保证给每个程序提供一种隔离的工作环境，这就是所谓的保护机制。所谓保护，从 CPU 角度来说，主要是指对存储器访问的保护；从运行在 CPU 上的操作系统角度来说，主要是指各个程序运行在不同的特权级别上。由于程序具有的权利和能力的不同，屏蔽了普通程序对系统、硬件、中断等的直接访问，

使得这些设备处于一种保护状态。

80386 CPU 提供了四个特权等级，通常称为 Ring0、Ring1、Ring2、Ring3。Ring3 特权级别最低，Ring0 特权级别最高。操作系统和 VxD 类型的设备驱动程序运行在 Ring0 级别上，普通应用程序（包括 DLL）运行在 Ring3 级别上。

保护模式的最大优点是：提高了系统的可靠性和健壮性，使得系统不会因为运行在 Ring3 上的应用程序崩溃而造成整个系统的崩溃。对于运行在 Ring3 上的应用程序，访问硬件作了一定的限制，避免了由于普通应用程序直接操作硬件资源而产生意外的错误。

保护模式的最大缺点是：不能像在 DOS 环境下一样直接访问硬件资源，而要通过设备驱动程序接口来访问。这使得用户在需要访问自己的硬件设备时，首先必须编写访问硬件设备的驱动程序。这也是我们要研究或者编写 Windows 环境下的设备驱动程序的原因。内存管理机制和中断处理机制是设备驱动程序开发人员必须了解的知识，下面简单介绍保护模式下的存储管理机制和中断处理机制。

1. 保护模式下的存储管理技术

存储管理技术实质上是指系统采用一种什么样的机制，将程序中用到的逻辑地址对应转换到实际的物理地址。

由前面已经知道，在保护模式下采用的是分段分页的存储管理技术。在了解分段分页的管理技术之前，先介绍几个地址概念。

● 逻辑地址

在应用程序中用来访问存储器的地址，就是逻辑地址。

使用这种地址的时候，不考虑地址在操作方式和实际物理硬件上的实现细节，程序通过 CPU 的存储管理机制，自动将逻辑地址转换为对存储器进行有效操作的地址。

在实模式中，逻辑地址通常称为分段地址，而在保护模式中，逻辑地址常称为虚拟地址。

● 虚拟地址

逻辑地址在保护模式中常称为虚拟地址。

实模式中的分段地址形式为：段基地址加偏移量。段基地址是由段寄存器中的数值被左移四位而得到的，段基地址是段的实际物理地址。

保护模式的虚拟地址形式为：段基地址加偏移量。段基地址是通过段选择器所提供的描述符而得到的。描述符中描述了段的基地址和一些控制信息，段基地址和偏移量相加得到线性地址。

● 线性地址

无论是实模式中还是保护模式中“段基地址加偏移量”形式构成的地址，最终都被转换成一个线性地址。线性地址是一个无符号的数，表明了在 CPU 线性处理空间中可以访问的存储器的相对位置。

对于实模式来说，线性地址就是物理地址。而对于保护模式来说，如果没有采用分页机制，那么线性地址就是物理地址。如果采用分页机制，线性地址还要通过分页部件的处理，将线性地址转换为实际的物理地址。

● 物理地址

物理地址就是在实际物理存储器中的位置，是逻辑地址经过存储管理机制作用后的结果。

在保护模式中，对于物理地址常采用片内两级管理，即分段管理和分页管理，支持容量极大的虚拟存储器。下面我们来讨论这两种管理机制。

1) 分段管理

保护模式下的虚拟地址是由两部分构成的：一个 16 位的段选择器和一个 32 位的地址偏移量。图 1.1 为虚拟地址结构图，图 1.2 为段选择器结构图。

图 1.1 虚拟地址结构图

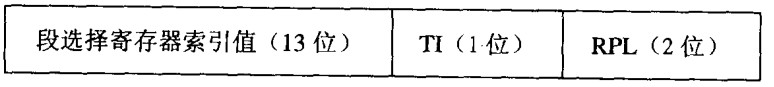

图 1.2 段选择器结构图

段选择器中存放的不是实模式中所说的实际物理地址，而是一个指向系统描述符表的指针。这个描述符表是由操作系统定义的，就是我们通常说的 GDT（全局描述符表）和 LDT（局部描述符表）。

描述符表是由操作系统自己来管理的存储在存储器中的系统数据结构，分为 GDT 和 LDT，整个系统只有一个 GDT 描述符表，而各个任务都有各自的 LDT 描述符表。

GDT 和 LDT 分别在 GDTR（全局描述符表寄存器）和 LDTR（局部描述符表寄存器）中存放了物理基地址、段界限、段的访问权限。

在图 1.2 所示的段选择器结构图中，段选择器索引值采用了 13 位，从而决定了地址空间可以分为 8192 个段。RPL 主要用于保护机制，它与存储器的地址选择没有关系，表示选择器的请求特权级别；TI 是指示位，用来确定段选择器指向的段是全局地址空间的段，还是局部地址空间的段，它与存储器地址的选择有关。

通过这样的分配，段选择器可以有 14 位来确定存储器中的每个段。存储器可以分为 16 384 个段。

地址偏移量是 32 位，因此每个段可以达到 4 GB。综合段选择器和地址偏移量，一个保护模式的应用程序可以访问 64 TB 的虚拟地址空间。

采用分段管理机制，使得保护模式的应用程序可以在很大的虚拟地址空间中操作，实现大容量存储器的管理。

2) 分页管理

前面我们已经讲过，应用程序中使用的虚拟地址经过分段部件处理后形成线性地址，如果不允许分页，那么此线性地址就是物理地址；如果允许分页，线性地址将经过分页处理后才能形成物理地址，实现对实际物理存储的访问和操作。

在 80386 以后的 CPU 中，管理内存的一个重要方法是采用分页管理，这是实现内存按需分配的基础，这种机制的物理基础是 CPU 的分页部件。在分页机制中，页面大小固定为 4 KB。

在实现分页管理过程中，需要进行页目录表和页表的查询，下面我们来介绍分页管理机制。

在系统中存放一个页目录表，页目录表的起始物理地址通常存放在 CPU 的控制寄存器 CR3 中。页目录表大小为 4 KB，包含 1024 个页目录项，每个页目录项对应一个页表，页表的地址是由页目录项给出的。一个页表的大小为 4 KB，共有 1024 个页表项，这样，一个页目录表可以映射到 4 GB(1024×1024×4 KB)大小的空间，即一个页目录表可以管理 4 GB 的空间。在上述表达式中，第一个 1024 表示一个页目录表中有 1024 个页目录项，第二个 1024 表示每个页目录项对应于一个页表，4 KB 是每个页表空间的大小。

4 GB 是分段管理中每个段可以管理的地址空间，这样，在保护模式存储管理机制中，对于每个段的空间管理，在分页机制中又由分页机制来管理，可以将这个段中的对应地址空间划分为页来管理，至于这个页的位置是在真正的内存中，还是在硬盘中，以及什么时候调入合适的页来工作，则属于操作系统的虚拟内存的管理技术。

上面的表达式中，一个待转换为物理地址的线性地址格式如图 1.3 所示。

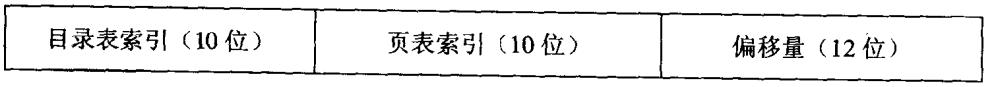

图 1.3 线性地址格式

其中目录表索引表示线性地址所在的页在哪个页目录项中，这个值乘以 4 得到的数值表示在目录索引表中的偏移量。目录项中给出的数值对应于页表的起始位置，是 20 位的物理地址；目录项表地址是由 CR3 寄存器给出的，是 20 位的地址。

页表索引表示此线性地址所在的页在哪个页表中，这个值乘以 4 得到的数值表示在页表中的偏移量。页表中给出的数值是物理地址的高 20 位，线性地址中的偏移量给出了物理地址的低 12 位，这样经过查表转换后，一个线性地址转换为一个 32 位的物理地址。至于这个物理地址的物理位置是在真正的物理内存中，还是在虚拟的物理内存中，是用操作系统来调度的，都与 CPU 的工作模式无关。后面我们还将介绍 Windows 系统中的虚拟内存管理技术。

线性地址到物理地址的转换过程可以用图 1.4 描述。

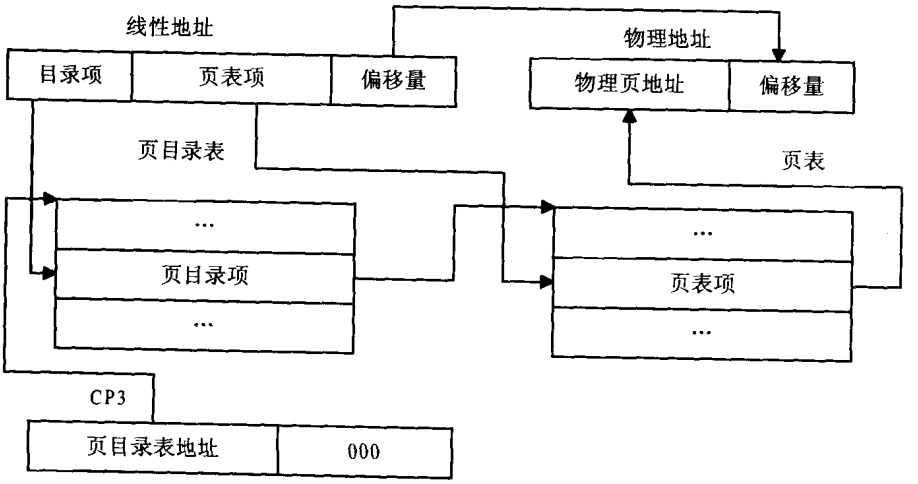

图 1.4 线性地址到物理地址的转换

2. 保护模式下的中断处理

保护模式下的中断处理方式和实模式下的中断处理方式很相似，都是通过响应相应的中断或者异常，从内存适当位置取出中断处理程序或者异常处理程序的起始地址，接着将控制权移交给处理程序，当处理程序完成处理后，再返回到原程序继续执行。

两者主要不同之处在于具体实现方式的不同：在实模式中断处理中，中断向量表总是位于物理存储器的低 1 KB 中，共 256 个中断向量，对应 256 个中断；在保护模式下，也有 256 个中断，但是中断向量是通过中断门或者陷阱门给出的，中断门或者陷阱门位于中断描述符表 IDT 中，中断描述表地址是由 IDTR（中断描述表寄存器）给出的。

1.1.3 虚拟 86 模式（V86 模式）

虚拟 86 模式是一种特殊的保护模式，是 CPU 为了既能向下兼容 8086 下的程序，同时也能获得保护模式的优点而采取的一种工作模式。

虚拟 86 模式中，CPU 虚拟生成多个 8086 处理器，使得在 DOS 下生成的程序能够运行在保护模式存储器中。这种模式在存储器分段管理方面和实模式一致，采用段基地址加偏移量方式，线性地址是由段寄存器中的物理地址值左移 4 位再加上偏移量组成。在分页管理方面，支持分页管理，但是寻址方式不同于保护模式中的寻址方式，只是将 1 MB 分为 256 个 4 KB 的页面。

这种模式在多任务系统中，可以同时运行多个 V86 程序，将不同的 V86 任务映射到不同的物理存储空间，给用户的感觉好像是多个 8086 在同时运行，从而产生虚拟 86 的概念。

1.2 Windows 9x操作系统

Windows 操作系统和其它操作系统一样，是个复杂的系统，其主要任务是管理和抽象系统资源。运行在操作系统中的客户程序通常是应用程序，包括 Windows 和 DOS 程序。操作系统对资源的管理，通常是指对内存、处理器工作时间、特定硬件设备（如显示器、键盘、鼠标、串口）和其它与操作系统相联系的硬件设备的访问和存取等。

从 Windows 9x 系统开始，操作系统是一种具有保护机制的操作系统，对系统底层操作采取了屏蔽策略。操作系统为每个应用程序提供了一个虚拟机，应用程序就运行在这个虚拟机中。这样一来，似乎应用程序在独占资源，使得应用程序和系统程序、其它程序的运行隔离开来，不会因为应用程序的破坏而影响系统程序的运行。

Windows 9x操作系统是分层操作系统，程序运行在系统中两个截然不同的层上。

(1) 操作系统上层，提供API服务，包括USER、GDI、KERNEL提供的服务。应用程序开发人员很熟悉系统提供的API服务，这些服务为建立用户应用程序提供了便利。

(2) 操作系统底层是VMM，即虚拟机管理器，它能产生和管理多个虚拟机。

在实际中，Windows 9x 系统不是单一的操作系统，而是操作系统的集合。在保护模式下有两种操作系统存在，即 VMM/DPMI，VMM 是虚拟机管理器，DPMI 是 DOS 保护模式接口。其中，操作系统的核心（KERNEL）是由 VMM（虚拟管理器）组成的。系统核心提供了 900 多个服务函数来管理内存，管理物理设备，处理中断，创建网络协议栈，管理文

件系统等，通过这些服务来完成操作系统的主要任务。

Windows 系统是多任务系统。系统控制权掌握在运行于 Ring0 层上的 VMM 和 VxD 中。系统中有两种虚拟机：DOS 虚拟机和系统虚拟机。所有的 DOS 程序都运行在 DOS 虚拟机中，一个 DOS 程序对应一个 DOS 虚拟机；所有的 Windows 程序都运行在系统虚拟机中，16 位的 Windows 程序共享同一个地址空间，而 32 位的 Windows 程序各自有独立的地址空间。

Windows 系统将 CPU 的时间轮流分配给每个虚拟机，这样，在各个程序之间是合作多任务工作，在虚拟机中是优先多任务工作。

Windows 9x 系统的内核基本一致，都是基于 DOS 内核的操作系统，这和 Windows NT 系统是不同的。Windows 98 系统比 Windows 95 系统增加了许多服务和支持功能，它不仅支持 Windows 2000 下的 WDM 驱动程序，而且也支持 Windows 95 下的 VxD 设备驱动程序。

为了更好地理解操作系统结构，下面分别来介绍有关概念。

1.2.1 虚拟机（VM）

虚拟机是指执行应用程序的虚拟环境，是用软件创建的系统资源假象，是系统资源的一种仿真。虚拟机一般可仿真以下系统资源：可访问的内存空间、输入/输出操作、外围设备、中断操作等。每个虚拟机都有自己独立的内存地址空间、I/O 端口地址空间、寄存器状态、堆栈、应用程序、系统服务程序、中断表状态和执行优先权等。虚拟机和运行在其上的应用程序通过这些环境因素再交互，就好像这个应用程序独占资源，运行在真正的机器上一样。

也可以把虚拟机理解为实际机器和应用程序之间的一个 API 函数。这个 API 函数不同于一般的 API，它是由中断、BIOS 调用和 I/O 端口组成的，主要用来完成应用程序与实际机器之间的交互。

如果 Windows 系统能以某种方法完美地实现这种 API 的作用，那么在虚拟机上运行的程序就如同真正在实际机器上运行一般。

Windows 9x 系统中，常采用 VxD（虚拟设备驱动程序）的方法来完成这种函数作用。同时，为了协调和监视各个虚拟机之间的工作，Windows 系统采用 VMM(虚拟机管理器)来分配任务和管理工作。

在 Windows 系统中，有两种类型的虚拟机：系统虚拟机和 DOS 虚拟机。所有的 Windows 程序都共享一个系统虚拟机。DOS 程序运行在 DOS 虚拟机中，每运行一个 DOS 程序，将产生一个 DOS 虚拟机，不同的 DOS 程序运行在各自的 DOS 虚拟机中。

虚拟机是由 VMM 创建的，每个虚拟机都有自己的虚拟机句柄和虚拟机控制块。虚拟机句柄实际上是指向虚拟机控制块的一个线性地址，用来唯一标志该虚拟机；虚拟机控制块是虚拟机的核心部分，是用来存放 VMM 和各种 VxD 管理虚拟机时使用的数据的一种系统数据结构。

每个虚拟机都有 8086（V86）组件、保护模式（PM）组件和一系列 VMM 维护的 Ring0 数据结构部分组件。

(1) V86 组件：虚拟机中的 V86 组件包含 1 MB 的地址空间，Windows 使用处理器的虚拟 8086 模式，在虚拟机中运行处于实模式状态的 DOS 应用程序。

运行在这个组件中的程序包括 DOS 程序，如 DOS 设备驱动程序和已经存在的 TSR 程序。

(2) PM组件：运行在虚拟机中的程序，可以使用DPMI进入到保护模式中，所有Windows应用程序都是保护模式的应用程序，也就是说，这些程序可以装入到扩展内存中，并且运行在 CPU 的保护模式中。

(3) Ring0 组件：虚拟机管理器VMM为每个虚拟机维持了一系列数据结构，包括Ring0上的堆栈结构，这个数据结构常用来在异常或者中断出现时，从应用程序传输控制信息到系统中。这个数据结构也包括一些调度信息和实例数据。

当控制信息从V86组件或者是PM组件传递到Ring0组件中时，在Ring0组件的堆栈结构中Client Register Struct数据结构，常常用来存储虚拟机CPU的信息。

1.2.2 虚拟机管理器（VMM）

VMM 是操作系统的核心，是一个 32 位的保护模式程序，主要任务是产生、运行、监控和中止虚拟机，同时建立和维护一个支持虚拟机的工作框架。VMM 也提供了一些服务，用来管理内存、中断和出错等任务。

在最初的操作系统设计中，VMM 是用来控制和管理系统中所有硬件资源，VxD 程序是 32 位保护模式的动态链接库，用来扩展操作系统的功能。在 VMM 管理和控制下，VxD 程序和 VMM 程序一起实现对系统硬件资源的管理，并且处理系统中的软件和硬件事件。

VMM 本身是一个 32 位的保护模式程序，是一个 VxD 程序，只不过这个 VxD 程序作用不同于其它 VxD 程序。作为其它 VxD 程序的监视器和控制器，这个 VxD 程序存放在操作系统的系统目录下的 VMM32. VxD 文件中。

为了更好地理解 VMM 在系统中的作用、地位等，我们先来看看 Windows 9x 系统的启动过程。

Windows 9x 系统的启动次序：

(1) 加载 io.sys；

(2) 执行 config.sys 和 autoexec.bat；

(3) 调用 win.com；

(4) win.com 运行 VMM32.VxD(VMM32.VxD 实际上是个简单的 dos 的 exe 文件)；

(5) VMM32.VxD 用 xms 驱动程序把 VMM 加载到扩展内存；

(6) VMM 初始化自身及其它默认的 VxD；

(7) VMM 把机器转入到保护模式并创建系统虚拟机；

(8) 最后被加载的虚拟外壳设备在系统虚拟机上通过运行 krnl386.exe 来启动 Windows；

(9) krnl386.exe 加载所有的文件，最后是 Windows 9x 外壳。

从上面的过程可以看出，VMM 虚拟机管理器是第一个被加载到内存的 VxD 程序，之后，通过它来创建系统虚拟机，并且初始化其它默认的 VxD 程序，VMM 对其它的 VxD 提供了许多的服务。

VMM 和 VxD 程序都是运行在 Ring0 层上的程序，具有最高的特权级别。同其它运行在 Ring3 层上的普通应用程序不同，这些程序平时是潜伏的，处于不被激发状态，只有当需要它们服务的事件发生时，才被唤醒。这种需要服务的事件往往是指硬件中断、错误异

常、特殊事件等。

VMM 虚拟机管理器是不可重入的，这就要求 VxD 程序的访问必须和 VMM 服务是同步的。VMM 和 VxD 具有很高的优先特权，因此在编写 VxD 程序时应该很小心，否则，当出现错误的时候是无法控制的。

1.2.3 虚拟设备驱动程序 VxD

虚拟设备驱动程序我们在第二章中将要详细介绍，在这里，简单地介绍一下关于 VxD 的有关问题。

Windows 9x 系统采取了底层屏蔽保护措施，应用程序对底层的操作必须通过运行在 Ring0 层上的设备驱动程序来实现。Windows 9x 下的驱动程序类型为 VxD。

VxD 是虚拟设备驱动程序的简称，其中 x 表示各种设备的名字，如虚拟键盘驱动程序（VKD），虚拟鼠标驱动程序（VMD），虚拟可编程中断驱动程序（VPICD）等。

VxD 本身是运行在 Ring0 层上的 32 位的可执行程序，它是 Windows 应用程序和实际硬件之间的接口，也是操作系统管理资源的接口，主要完成对设备的虚拟化功能。

Windows 系统是多任务多线程的操作系统，外部设备往往同时被基于 DOS 或者 Windows 的多个线程共同使用，这种程序和设备之间的对应关系不同于 DOS 环境下程序和设备一一对应的关系，而是“一对多”的关系。这样，要使得多个线程使用外设，必须实现设备的虚拟化，让应用程序感觉到每时每刻自己都是独占硬件设备。

设备的虚拟化是通过运行在 Ring0 层上的设备驱动程序来实现的。驱动程序充当了实际硬件的“替身”，线程或者应用程序总是与硬件的“替身”（即我们编写的 VxD 驱动程序）交换信息。

以上说明了 VxD 驱动程序的一个重要的用途是实现设备的虚拟化，让多个线程可以访问实际硬件。但不要理解为 VxD 必须和实际硬件连接在一起。更确切地说，可以把 VxD 看作是一个运行在 Ring0 层上的 DLL，任何想在 Ring0 层完成的工作，都可以编写 VxD 程序来实现。这样，如果 VxD 程序没有与很具体的硬件连接的话，可以把 VxD 看做你自己程序的一种功能上的扩展。

最后，还需要注意几点：

(1) 设备驱动程序 VxD 是 Windows 9x 系统特有的驱动程序模型，它不适合 Windows NT、Windows 2000 系统，因此，不能将驱动程序进行平台移植。

(2) VxD 程序是运行在 Ring0 层上的特权级别程序，可以对系统做任何事情，一个恶意的 VxD 程序可以破坏掉这个系统，而对于这种恶意的 VxD，系统没有任何的保护措施。

(3) 通常情况下，在使用 VxD 的时候，一定要三思，看是不是必须采用 VxD 的方法，如果没有其它的 Ring3 层的方法，再使用 VxD 方法；否则，可考虑采用其它方法。

在第二章中，我们还要详细介绍 VxD 虚拟设备驱动程序。

1.2.4 Windows 9x 系统的基本构架

为了更好地理解 Windows 9x 系统结构，在这里我们简单地介绍这个系统的基本构架。Windows 9x 系统是一种分层的结构，层次如下：

(1) 在系统最底层为 DPMI 服务器，它和 VMM 构成了操作系统的核心。VMM 是建立

在 DPMI 之上的，系统启动时，DPMI 先于 VMM 启动。DPMI 是 DOS 的保护模式接口，最初是用来将 DOS 程序切换到保护模式下，并实现大于 1 MB 内存的使用。Windows 系统是基于 DOS 核心的系统，在初始化时利用 DPMI 来完成初始的实模式到保护模式的转换。

(2) 其次是虚拟管理程序，即虚拟机管理器（VMM）。这就是我们通常所说的 Windows 系统的最底层。

系统中并没有 VMM.VxD 这样的程序存在，它被包含在 Win386.EXE 或者 VMM32.VxD 这样的程序中。管理维护系统虚拟机工作环境包括线程之间的切换。

(3) 第三层是各种虚拟设备驱动程序，包括*.386、*.VxD 和其它具有 VxD 文件头的程序。

这些程序运行在 Ring0 层上，在 VMM 启动之后被启动，并和 VMM 构成 GUI（图形用户界面）子系统运行的运行基本环境。

VMM 是一个可扩展的操作系统，通过系统本身提供的 VxD 程序和用户自己设计的 VxD 程序来完成整个系统的控制任务。这一层实质上是 VMM 功能的扩展，是系统不可缺少的一部分。

(4) 各种*.drv 驱动程序。这些程序和 KRNL386.EXE、USER.EXE、GDI.EXE 等组成了 GUI 和底层的通道，这样形成了 Windows 系统的基本系统。这些程序是运行在 Ring3 层上的程序。

(5) 各种扩展程序。这些扩展程序包括 OLE、WINSOCK、RPC 等，扩展程序已经不再和底层打交道，而是被各种 DLL 程序所掩盖，这样完全进入了 GUI 系统。

Windows 的底层被掩盖，用户无法直接操作系统硬件，而是通过各种驱动程序来操作。

(6) 在 Windows 3.11 系统中，有了 Windows 95 的一些基本特征，32 位文件系统出现。32 位文件 API 的出现，使得 32 位操作系统的雏形形成。

(7) Windows 95 是 Windows 3.11 功能的扩展，最大特点是增加了长文件名和动态 VxD 的装载，在 Windows 3.11 下，VxD 只能静态装载。

(8) Windows 98 系统除了支持 Windows 95、Windows 3.1 下的驱动程序外，一个很大的特点就是支持 WDM（Windows 32 Device Model）。

WDM 设备驱动程序模型是基于 NT 的，是 Windows 2000 下的驱动程序模型，也是 Windows 系统发展的趋势，系统资源越来越多地被操作系统控制，要控制资源必须通过操作系统的控制。

1.2.5 内存管理技术

在了解 Windows 操作系统基本结构基础上理解操作系统的内存结构是理解操作系统、深入到操作系统核心的关键。下面我们从 Windows 操作系统的内存结构和内存管理两个方面来了解 Windows 系统的内存管理技术。

1. 内存结构

VMM 采用 Intel 86 或者更新处理器的分页分段内存管理技术，为系统虚拟机建立了 32 位的虚拟地址空间，操作系统需要管理的内存空间为 4 GB，如何管理好这 4 GB 的空间是内存处理技术的关键。