第一章 概述：Windows、建网以及 NetWare 系统

当业务量面临经济现实和采用尽可能有效方式完成所做工作时，就要将它们转到提供有关经济系统的易用接口的解决方案。使用 Microsoft Windows on Novell NetWare 系统往往会吻合许多业务需求的这种解决方案。组建网络正在改变计算机工业的面貌，因为它提供了共享资源和共享信息的高性能、低成本解决方案。NetWare 是最高性能的和最可靠的建网解决方案。

NetWare 系统允许通过多机种桌面环境来共享数据和资源。每个用户或每种业务能够选择所要求的桌面系统，并同另外的桌面系统进行通讯。许多人正选择 Windows 作为桌面方案。利用 Windows 可提供一种更直观的图形用户接口(GUI)。这个接口允许人们更快地一次学会使用新的软件，并且通过使用符号、图象和辅助软件更方便地掌握那种知识，以便寻求和运行应用软件(本书中的应用就指应用软件或应用程序——译者)。NetWare 和 Windows 两者都是很好的系统，而且应用也需要运用两者的力量。加强 NetWare 和 Windows 能力的应用是未来计算机技术的出发点，并将对众多的业务需求提供解决方案。本书中所给的步骤会将你引导到利用 NetWare 系统所感知的 Windows 应用机会之窗口。

1.1 Windows

由 Microsoft 公司开发的 Windows 操作系统是一种带有图形用户接口(GUI)的非优先操作系统。这个接口通过组织图标、菜单和应用窗口方面的特点而便于使用。每种应用都使用一个图标来选择和运行它，使用一个菜单来选择参数，而使用一个窗口用于输入和输出。

使用 Windows 时，一个用户往往必须开发一个完全新的符号集。在这个符号集中包含的内容是术语和简略词，诸如动态链接库(DLLs)、事例、发射(Launching)、焦点(focus)、段管理、定义文件、输入、输出、序数、输入库(implibs)、链接文件、符号文件等。

一个**动态链接库**是这样一种库，即在运行时链接其包含的代码。通常被称为一个 DLL，这些库允许相同的代码被几个应用所使用，这样就减小了个别应用的规模(长度大小)。DLL 的另一个优点是，当库中每次修复一个错误时，不需要重新链接一个应用。

事例系指每次执行一个应用。若一个应用被执行多次，就存在该应用的多个事例。所有 Windows 应用都拥有一个事例句柄(handle)，它也被称为一个任务 ID。当执行一个应用时就发射它。

焦点是一个使用的术语，它指出哪些应用或窗口当前是发生作用的(激活的)。激活的窗口将拥有处理焦点。

段管理对于 Windows 体系结构中的应用程序是必要的。由于应用运行在保护方式，代码段是只读而数据段是读/写。这些段的目的是不相同的。因此，数据和代码管理也要求判定，它们将存在于哪些段以及这些段的特征将是什么。这些段可被定义为诸如 LOADONCALL、PRELOAD、FIXED、MOVEABLE、DISCARDABLE、MULTIPLE 和 SINGLE。在应用定义文

件中定义这些段。定义文件中也定义程序标题、程序类型、输入函数和输出函数。

输入函数是应用知道在别处存在的函数，这里的别处是指另外的应用或 DLL。这些函数通过指定定义文件(后跟一个句点)的 IMPORTS 分段中的源(应用)名以及该函数名进行输入。当函数在 IMPORTS 分段中被指定时，链接程序并不在链接时刻从库中寻找它们。

输出函数是从一个应用输出的函数。通常，指定其名字后，它们后跟一个@及一个数字。这个数字被称为序数(ordinal number)。

序数允许 Windows 系统通过代替扫描源文件中的名字的数字来快速引用一个函数。当应用程序 IMPLIB. EXE 运行於指定文件上时，一个输入库从定义文件中指定的输出进行组建。当链接一个应用让链接程序知道能在何处找到这些调用时，此时就能使用这个库。定义文件被设置在 Windows 应用的链接行或者在链接文件内。

链接文件定义将被链接成为可执行文件或 DLL 的所有目标文件(.OBJs)。链接一个程序就生成一个映像文件，从映像文件建立一个符号文件。应用程序 MAPSYM. EXE 运行在将它变换为一个符号文件的映像文件上。

1.1.1 Windows 应用编程概述

Windows 应用程序是报文(亦称消息)(message)驱动的。一个报文驱动系统是这样的系统，在系统中应用根据被传送到报文队列的一个报文发生作用。在 Microsoft Windows 中，将报文从操作系统传送到应用的结果就是要做这样的操作：移动光标、显示一个键入的字符、拉出一个窗口等等。该应用要确定需要哪些报文，以便在这些报文上进行操作和产生作用。若队列中的报文并非是应用所期待的，那么该应用必须将报文传送出去。为此，一个 Windows 应用将发布一个调用对：TranslateMessage()和 DispatchMessage()。

Windows 报文是这样的数据结构，它指出报文来自何处以及其值大小。这些结构的句柄从应用队列移动到应用队列。用这种方式，每个应用就可查看从系统进入的报文。Microsoft 提供一个函数 DefaultWindowsProc，它允许应用以标准的形式来处理标准的报文。例如，若一个应用仅希望处理某个字符，它能将所有别的键盘报文传送给缺省(亦称默认，default)过程。当鼠标器移动光标时，这个缺省过程将在屏面上移动光标。

应用开发人员并不需要学会移动光标所必要的所有功能。计算机上的每种硬件作用都会产生 Windows 系统下所传送的报文。移动鼠标器就产生几十个报文。为了查看产生多少报文及其内容，Windows 已在 Software Development Kit (SDK，软件开发套件)中提供一个称为 SPY 的工具。

由于 Windows 是报文驱动的，必须记住，Windows 是属于非优先级的。换句话说，Windows 将并不产生一个应用程序的执行，该应用必须放弃对 Windows 操作系统的控制。在 Windows 中应予以避免的一种编程结构就是紧凑循环。例如，当紧凑时，while 循环像如下形式：

```
while (inUseFlag);
```

它被用于 Windows 应用中，将使那个应用完全消耗 CPU 周期。决不会放弃对操作系统的控制，该应用将出现挂起。事实上，在控制的程序中带有紧凑循环，就没有报文能被任何应用程序所处理。

1.1.2 Windows 应用存储器

Windows 允许应用使用机器中的全部存储器,不仅在 1Mb(兆字节)以下。Windows 中的内存管理机构处理程序段从一个位置移动到另一位置,以便拥有最大可用自由存储器块。这样,就发挥了 Windows 操作系统的最大优点:一个应用仅当绝对必要时才声明(说明)被固定的段。必须要有固定段的例子,包括异步地从一个硬件中断或者从 Windows 定时器进行访问的过程。若该应用将 MakeProcInstance 用于需要异步调用的过程,就不需要固定段了。这样一个过程对于下列情况是最优化的,即其目标是传送报文给应用程序,以便指出应发生什么作用。

在 Windows 内存管理机构中,程序数据可从局部存储器场(亦称库,pool)或全局存储器场得到分配。局部存储器场被设置在程序的堆阵(heap)上。全局存储器场由 Windows 操作系统管理。

在 Windows 下,相同的应用可以多重执行。鉴于这个因素,应用程序必须通过声明它为多重的,才能援引(说明)它们的数据段,或者使用一个表来援引该数据。然而,一个动态链接库(DLL)仅能拥有一个数据段,而必须做到下列三件事情之一:避开全局数据,用表说明全局数据或者使用由应用传送的数据缓冲器。

Windows 使用句柄来允许被分配的存储器进行移动,并仍然可访问该应用程序。当存储器由局部堆阵或全局堆阵进行分配时,在再引用它之前应封锁它。否则,当数据段的物理位置变化时,变量会发生改变。

1.1.3 Windows 操作方式

Windows 3.0 中存在三种操作方式:增强的(Enhanced)、标准的(Standard)、和实址(Real)方式。在 Windows 3.1 中仅存在两种方式:增强的和标准的。实址方式需要一个基于 8087 的计算机或更高级计算机。

实址方式运行于 80286 或 80386 机器上的实址方式仿真状态。在实址方式中,DOS 应用的内存和 Windows 应用的内存全部存在于 1Mb 内存边界之下。这就是能由段:偏移寻址方式被引用的全部内存。当 Windows 应用正在执行时,DOS 应用就不能执行,反之亦然。在 Windows 3.0 以上的任何版本中将不支持这种方式。

标准方式要求至少一台 80286 计算机。这种操作方式使用 80286 指令系统。Windows 应用运行于 80286 保护内存中。任何应用,若试图写入并不占有的内存,将导致一个不可恢复的应用错误或者总的指出违反内存保护的保护故障。再有,像实址方式中一样,当 Windows 应用正在执行时,DOS 应用不能执行,反之亦然。贯穿于本书,当讲到标准方式时,由于两种方式中的操作相似性,实址方式和标准方式两者都被引用。增强方式要求至少一台 80386 计算机。这种方式运用 80386 指令系统和虚拟机能力。

1.1.4 虚拟机器

一台虚拟机器(VM)是一种存储器影像和寄存器值的集合,当执行时,它好像是完整的计算机,但只是所有运行在相同时间或轮回的几个虚拟机器之一。轮回处理的每台虚拟机的过程被称为时间分片(time-slicing)。当操作系统管理每台虚拟机实现这些轮回时,该系统被称为任

务交换(task switching)。任务交换表示该系统从任务转换到任务。

一台虚拟机正确认为是虚拟的而不是物理的。这种虚拟机原理已在大型主机上使用很长一段时间了,它允许众多用户访问物理系统资源。在推出 80386 芯片之前,基于 Intel CPU 芯片的计算机中,要拥有多台虚拟机的能力是不可取的。80386 芯片允许操作系统使用诸如虚拟机的原理来进行编写。

在 Windows 下,所有 Windows 应用和 Windows 核心运行在 VM1 或系统 VM 中。每个 DOS 提示符运行于分离的 VM。例如,若你正用 Windows 运行程序管理程序,且拥有两个 DOS 程序在运行,每一个来自一个分离的 DOS 提示符,那么你就拥有三台虚拟机。

1.1.5 虚拟机管理程序(Manager)和 DOS 保护方式接口

在 Windows 下面,DOS 提示符优先运行,就好像 DOS 正常做的那样。这意味着任何过程可在允许中断的任意时刻被中断。为了从 Windows 仿真 DOS 环境,必须要管理输入和输出。为了管理来自键盘、鼠标器、屏面或其它硬件的信息输入和输出,要编写虚拟设备。一个虚拟设备是一个运行在 386 保护方式中环(ring)0 的设备驱动程序。一个虚拟设备运行在 Windows 操作系统的一个层次下面,该层次称为虚拟机管理程序(Virtual Machine Manager,略写为 VMM)。

VMM 许可 Windows 对每个 VM 的中断、存储器和通讯进行虚拟化。这种虚拟化就是每个虚拟机好像是一个独立过程在运行的理由。每个虚拟设备作为 Windows 中的应用和计算机硬件 设备之间的多路复用层在发生作用。虚拟设备允许所有的应用向每个设备通讯而避免冲突和死锁现象。在一个虚拟机中某个内存单元的信息不一定是在另一虚拟机的相同位置。因此,一个虚拟设备必须要知道哪个虚拟机正在运行,并将正确的信息写到仅对那个虚拟机的内存中的正确单元。

由于 Windows 应用运行于保护方式,编写一个接口允许它们同运行于实址方式(8088 方式)的 Windows 之前被装入的驱动程序进行通讯。这个接口被称为 DOS 保护方式接口(DPMI)。在 DPMI 下,应用可分配全局 DOS 存储器、分配一个回调(call back)入口点、分配引用 Global(全局)DOS 存储器的描述符,并用一个远程返回栈结构(stack frame)或一个中断返回栈结构来模拟实址方式调用。在 DPMI 下被分配的回调用每个 VM 的最大值 16 来实现。因此,一个试图从实址方式进行异步访问的应用将需要一个虚拟设备,不是一个 DPMI 回调,因为正确的 VM 并不会运行在异步回调被执行的时间。

Windows 操作系统是时间分片的,而不是多任务的。这表明运行在后台的所有应用同运行在前台的应用轮回使用 CPU。一个标记成仅运行于前台的 DOS 提示符将消耗所有在后台的处理。当应用继续进行处理时就发生后台处理,虽然它并非是当前激活的(发生作用的)程序。

在当前激活窗口的含义上,前台处理就会发生在一个应用中。计算机屏面上最主要的窗口就是运行于前台的应用。当标记为排斥(explusive)的 DOS 提示符运行在前台时,它不允许任何别的应用在后台执行。仅当 DOS 提示符对正确运行应用绝对必要时才将它标记为 Exclusive。当 DOS 提示符运行在排斥方式时,异步处理数据的应用不应同时运行。

当 Windows 装入时,先于 Windows 被装入的所有驻留应用(TSRs)被认为对所有 VM 是全局的。在 Windows 运行之后被装入的任何应用对那个 VM 就是局部的。为了举例说明全局

设备中被包含的数据，一个虚拟设备必须知道该数据位于何处。要知道有多少数据将被引证说明，并向 Windows 核心发出命令，在 Windows 初始化基础上虚拟化这个存储单元。

为了组建运行于 Windows 3.0 和 Windows 3.1 的应用，每个 C 文件必须包含 #define WINVER 0x0300 定义。此外，RC.EXE 实用程序需要用一个 V 3.0 兼容性转换来运行。例如：

```
rc  -30  ipxchat.rc
```

1.2 建网

建网允许用户通过网络上桌面机相互通信和桌面机与服务器通信而共享资源。这种通信通过硬件和软件的组合来实现，硬件的组成包括机器之间的某种电缆连接和使用网络接口卡。这些卡负责基于 ethernet、arcnet、token ring 等等通信协议的通信。软件由驱动程序组成，驱动程序从网络卡取出报文分组并将它们同运行于桌面机的应用和操作系统放在一起。运行于桌面机的网络软件通常被称作为客户(client)。位于网络上提供服务或资源的机器被称为服务器。本书后面将讨论一台客户机也能作为服务器发生作用。

在网络中，由应用所做的工作部分发生在客户机上，诸如信息显示和数据计算；而发生在服务器上的工作部分，则包括诸如将信息保存到磁盘及打印等。这种在拥有自己的中央处理单元的多台机器之间的劳动量划分被称为分布处理。当所有的工作在一个地方完成时就指集中处理。不位于网络上的一台个人计算机和一台大型主机是集中处理的两则例子。网络层以分布处理为基础的，而主机环境是以集中处理为基础的。采用个人桌面机的分布处理，其成本要小于集中处理所需求的大型多用户机器。

Windows 用户主要被设置在网络上。使用这些事实编写的软件将全面受到欢迎，而不会欢迎或者基于 Windows 的或者网络意识的软件。

1.3 NetWare 系统

网络市场的最大市场份额是由 Novell 公司的 NetWare 系统保持的。NetWare 网络软件被看作支持用户硬件平台选择的一个开放平台。NetWare 系统的应用当前仅受到人们想像力的限制。

1.3.1 NetWare Windows 驱动程序

NetWare Windows 网络方案包括一组标准的调用，任何 Windows 网络驱动程序都应支持这组调用。这些函数由 Microsoft 所确定，并在 Windows Driver Development Kit (Windows 驱动程序开发套件)中可找到。这些调用是支持调用 Windows 操作系统来实现网络状态的排队。对于 NetWare 客户，Novell 编写了一个驱动程序，称为 NETWARE.DRV。这个驱动程序提供了 Windows 所要求的用于网络驱动程序的接口。此外，NETWARE.DRV 提供了 Windows 应用的接口，使它访问网络和网络信息。NETWARE.DRV 是实现如下网络功能的 NetWare 驱动程序，诸如映像驱动器、注册到服务器、打印处理和接收网络报文等。这个驱动程序结合 shell(NETX.COM)或 NetWare DOS Requester (VLM.COM)一起工作，以便保持文件服务器连接和访问网络资源所需的其它信息。组成 NetWare Windows 支持的软件成份清单示如表 1-

1 所示。

表 1-1 为 NetWare 提供的 Windows 支持清单

驱动程序	版本
NETWARE. DRV	2.00
VNETWARE. 386	3.00
VIPX. 386	1.40
VLM. COM	1.02
*. VLM	(随 VLM. COM 一起提供)
NETX. COM	3.32
TBMI. COM	1.20
TASKID. COM	1.20
IPX. OBJ	3.10
IPXODI. COM	1.30
LSL. COM	2.00
TBMI2. COM	3.00

编写的其他两个驱动程序用来实现同 IPX. COM 和 shell(外壳)或者 NetWare DOS 请求程序(requester)的通讯。这些驱动程序分别是 VIPX. 386 和 VNETWARE. 386。VNETWARE. 386 实现从 DOS 提示符到 shell 的 int 2Fh 输入，并保证那个信息被返回给正确的 VM。VNETWARE. 386 是 386 增强方式虚拟设备驱动程序。在增强方式 Windows 中，所有 Windows 应用在相同的虚拟机 VM1 中运行。当希望一个 DOS 提示符时，用户转换到 DOS。由用户打开的每个 DOS 提示符都是一台独立的虚拟机。每台虚拟机将拥有它自己的存储器映像。对于相应虚拟机所给定的正确响应来说，VNETWARE. 386 保持每个请求的虚拟机信息。当用 DOS 提示符的一个应用向网络给出一个请求，诸如打开文件(Open File)请求，VNETWARE. 386 将截获那个请求，并在将它传送给 shell 或 Requester(请求程序)之前，将记录诸如虚拟机 ID 这样的信息。然后将请求传送给 shell 或请求程序，它们又把请求发送给服务器。在处理请求之后，服务器将回答信息送往工作站。shell 或请求程序然后设法返回这个信息。对于将被返回给正确虚拟机的信息来说，VNETWARE. 386 必须激活正确的虚拟机，然后拷贝该数据。运行每种 Windows 方式的必要软件在表 1－2 中归纳。

表 1-2 NetWare Windows 驱动程序及其发挥作用的方式

NetWare Windows 驱动程序	增强	标准	实址
NETWARE. DRV	X	X	X
VNETWARE. 386	X		
VIPX. 386	X		

NetWare Windows 驱动程序	增强	标准	实址
TBMI.COM*		X	X■
TASKID.COM*		X	X
TBM12.COM**	X	X	

注:* 仅对 Windows 3.0

** Windows 3.0 和 3.1

NETWARE.DRV 也向网络提供一个 Windows 应用的汇编程序接口。这个接口是通过一个函数调用 NetWareRequest 来提供的,它通常利用 DOS 下的 int 21h 来仿真调用。这些调用用于同使用 NetWare 核心协议(NCP)的 NetWare 文件服务器进行通讯。这是一个建立在 Novell 网际分组交换(IPX)顶部的同步协议。NCP 是一个认可推出的面向连接的协议。

一个请求是由 NetWare shell 或请求程序送往文件服务器的。shell 或请求程序然后等待来自服务器的回答。当这个回答来到时,shell 将该信息传回给发出请求的应用。请求用 IPX 上的报文分组进行传送和接收。IPX 是一个无连接数据报(datagram)协议。IPX 也是今天市场上最快的协议之一。由于这种高性能,shell 或请求程序能够用 NCP 很快地和有效地进行通讯。一旦信息进入工作站存储器中,用户可操作那个独立于文件服务器的信息。

这是网络和分布处理的最大优点之一。利用合适的工具和信息,可编写应用软件来利用网络上分布处理方面所得到的巨大潜力。这些工具包括相应版本的驱动程序和 NetWare C—Interface for Windows SDK。本书中所讨论的原理是一般的 NetWare 编程概念,并能通过 NetWare 应用编程接口(API)应用于其他的平台。

NetWare Windows 成份包括 NETWARE.DRV、VNETWARE.386、VIPX.386、NETX.COM 或 VLM.COM、TBMI.COM 或 TBM12.COM、TASKID.COM、IPX.COM 以及可用于同 NetWare 操作系统接口的 DLL。这些 DLL 可包括 NWIPXSPX.DLL、NWACCT.DLL、NWAFP.DLL、NWBIND.DLL、NWCONN.DLL、NWFILE.DLL、NWDIR.DLL、NWFSERV.DLL、NWQMS.DLL、NWSAP.DLL、NWDIAG.DLL、NWWORK.DLL、NWPSERV.DLL、NWCORE.DLL、NWMISC.DLL 和 NWPRINT.DLL,或者它们可包括 NWIPXSPX.DLL、NWNETAPI.DLL 和 NWPSERV.DLL。这些 DLL 组成了 NetWare C—Interface for Windows。在你的应用中使用 C—Interface 调用将许可你的应用成为 NetWare 意识的,并利用由 NetWare 提供的最好的服务和能力。DLL 另外也能通过 NetWare Client(客户)SDK 来提供。由客户 SDK 提供的 DLL 是 NWCALLS.DLL、NWNET.DLL 和 NWIPXSPX.DLL,这些 DLL 随 NetWare 4.0 的支持软件一起供货,当前从 Novell 公司可得到。

1.3.2 VIPX

在 Windows 增强方式下,内存实现虚拟化,系统利用计算机中的全部内存。Windows 应用所使用的内存通常要超过 1MB(兆字节)。另一方面,IPX.COM 是一个实址方式 TSR,它仅能引用 1MB 以下的内存。

为了允许应用从保护方式使用 IPX,就编写出 VIPX。VIPX 负责从保护方式应用将事件控制块(ECB)和分组缓冲器拷贝到全局 DOS 存储器,一方面保证仅当正确的 VM 发生作用时

才拷贝报文分组，而另一方面将 IPX 的调用从保护方式转移到虚拟 86 方式。虚拟 86 方式是指允许 DOS 应用进行处理的计算机运行方式。

为了在全局 DOS 存储器场中保留尽可能多的存储器，VIPX 仅在一个需求基础上分配低位存储缓冲器。对于发送分组来说，低位存储缓冲器立即被分配，而当发送完成又很快释放。然而，对于监听分配来说，VIPX 使用一个延迟分组分配过程。这表明几十个保护方式监听分组可以是未完成的，而在监听分组实际被 IPX 接收之前不会使用全局 DOS 存储器场的一个字节。

当 IPX 接收一个分组时，它首先查看在那个软插口(socket)队列上是否存在任何记入的未完成监听分组。若不存在未完成的监听 ECB，IPX 根据指定软插口上监听缓冲器的请求调用 VIPX。若 VIPX 拥有这样一个监听，VIPX 分配其上的全局 DOS 场的存储器被用于启动和被传送给接收此分组的 IPX。在这个 ECB 中的事件服务子程序(ESR)被设置成 VIPX 的自己的子程序。

当监听 分组被拷贝到全局缓冲器和事件服务子程序被调用时，VIPX 此时将全局存储缓冲器的信息拷贝到客户应用的缓冲器并释放全局存储缓冲器给其它监听分组使用。由于这种算法，VIPX 很少会有多于两个或三个存储缓冲器在任何时刻未完成。这样，对于 NetWare Windows DLL 的所有调用通过增强方式的 VIPX 进行处理。在实址和标准方式中，就必须要有不同的解决办法，因为虚拟设备仅运行在增强方式。

1.3.3 TBMI 和 TBMI2

对于实址和标准方式的解决方法是 TBMI.COM 或 TBMI2.COM，它们是对于 IPX 的任务转换缓冲器管理程序(Task-switch Buffer Manager for IPX)。这个应用是一个 DOS TSR，它在 Windows 之前被装入，并在任务转换期间缓冲应用的 IPX/SPX 分组。

在一个当前不发生作用的任务中的应用存储器不可异步用于标准方式中。例如，若 IPX 试图写入由 Windows 应用所交付的一个缓冲器，而一个 DOS 应用正从 Windows DOS 提示符运行，那个存储器单元将受到破坏。那个存储器并不被转换出来的应用所占有，该应用甚至可以是当前执行的 DOS 应用的代码段。

当使用 IPX/SPX 的 DOS 应用并非是激活的任务时，该转换也是成立的。为了在该应用再次激活之前访问这个结果和缓冲 IPX/SPX 交通流，Novell 曾编写了 TBMI.COM，TBMI2.COM 和 TASKID.COM。TBMI 的 1.0 版仅从 DOS 提示符支持 IPX/SPX 交通流。TBMI 1.1 版进行了修改以便结合 NWIPXSPX.DLL 一起同样用于缓冲 Windows 应用的报文分组。若用户仅运行 Windows 应用，就不需要 TASKID。然而，若用户曾计划使用一个 DOS 提示符，那么在发射使用 IPX/SPX 的一个 Windows 应用之前，TASKID 应运行于 DOS 提示符以及用户应转回到 Windows 而不用关闭 DOS 提示符。这将许可 TBMI 正确运行和识别用户何时已转换脱离它们的 Windows 应用。TBMI2.COM 2.0 版支持 Windows 3.1。TBMI2.COM 3.0 版支持 Windows 3.0 和 Windows 3.1 两者。当在 Windows 3.0 下运行 TBMI2.COM 时，在 NET.CFG 文件中要放有 Using Windows 3.0 = ON。如前所述，你此时将运行 TASKID.COM。在 Windows 3.1 时，仅需要 TBMI2.COM。TBMI2 及其同 NWIPXSPX.DLL 的交互作用将进一步在关于 IPX 一章中进行讨论。在本书的其余部分，所有参考引用将是针对 TBMI2，但将适当地应用于 TBMI 和 TBMI2 两者。

1.3.4 NetWare C—Interface for Windows

NetWare C—Interface for Windows 向应用开发人员提供大量的功能。这些 DLL 向 Windows 应用提供对网络的访问。这个访问类似于向 DOS 所提供的访问，且在 Novell NetWare C—Interface for Windows 软件开发套件中提供有资料。在本书中讨论的应用是利用 SDK 编写的，并将介绍如何使用其中包含的若干调用。对于寻求编写其应用所必要的功能的开发人员来说，表 1—3 就表示功能性被包含在哪些 DLL 内部。

表 1 3 NetWare DLL 和支持的功能性

DLL	功能
NWIPXSPX. DLL	IPX 和 SPX 函数调用 服务通告函数调用 诊断服务函数调用
NWNETAPI. DLL 或 NWCALLS. DLL	记帐服务函数调用 AFP 服务函数调用 装订库服务函数调用 连接服务函数调用 文件系统目录服务函数调用 文件系统服务函数调用 文件服务器环境函数调用 报文服务函数调用 各类支持函数调用 名字空间服务函数调用 打印和 QMS 服务函数调用 同步服务函数调用 交易跟踪系统服务函数调用 工作站服务函数调用
NWPSERV. DLL 或 NWPSRV. DLL	打印服务器服务函数调用。

在 C—Interface for Windows SDK 1.3 版中，NWIPXSPX. DLL 具有像 NWSAP. DLL、NWDIAG. DLL 和 NWIPXSPX. DLL 一样的功能。NWNETAPI. DLL 具有像下列 DLL 相同的功能：NWACCT. DLL、NWAFP. DLL、NWBINDRY. DLL、NWCONN. DLL、NWCORE. DLL、NWDIR. DLL、NWFILE. DLL、NWFSERVER. DLL、NWMESSAG. DLL、NWMISC. DLL、NWNAMESP. DLL、NWPRTQUE. DLL、NWSYNC. DLL、NWTTS. DLL 和 NWWRKSTN. DLL。NWPSERV. DLL 拥有像以前一样的相同功能。在 Novell NetWare Client SDK V 1.0 中，NWCALLS. DLL 包含 NWNETAPI. DLL 的功能性。NWIPXSPX. DLL 是带有两个附加调用的相同 DLL。NWNET. DLL 是一个新的 DLL，它支持 NetWare Directory Services（NetWare 4.0 版和新功能）。NWPSERV. DLL 已被 NWPSRV. DLL 所代替。

所有的 DLL 使用 Novell 所定义的版本格式。这种版本格式可以被 NetWare 的 VERSION. EXE 程序所读出。此版本格式由一个串组成，其中包含用混合字形的字说明，后跟等号和版本信息。例如，对于 NWIPXSPX. DLL 的版本串像如下所示：

versionString = "VeRsIoN = V. 1. 30 NWIPXSPX. DLL".

扫描这个版本串的可执行文件的示例代码由 Novell NetWare C—Interface for Windows 所提供。利用这个串的任何应用可拥有它的被 VERSION. EXE 读出的版本。实现寻找这个串的代码的任何应用可以用方便而标准的方法来检验 DLL 的版本。

本书中的应用是所编写的小型示例应用，它们利用了 NetWare 操作系统和分布计算的能力。能够加以实现的附加应用包括如下：

- 独立的记帐服务器——部门使用网络资源的付费系统；
- 授权服务器——控制电子文档的分级批准过程；
- E—mail 服务、打印服务、使用 QMS 或 SPX 的传真(Fax)服务器；
- 可进一步推动办公自动化的其它应用。

第二章　网络安全性：装订库和受托者控制

最近三年的刊物中，安全性问题已成为主流问题。破坏者使用系统上的后门和安全性空档来访问分类的和敏感的信息，同时破坏重要的研究数据。通过装订库(bindery)正确的使用NetWare安全性特点将帮助网络管理员避免安全性损害，这种损害可危及公司的数据和知识权益。

网络安全性由控制文件服务器访问的三个基本级别组成。这些级别由装订库、注册控制、以及文件的目录访问组成。这每一种级别结合其它措施可一起工作实现网络安全以及避免遭受侵扰和数据窃听之危险。

2.1　装订库概述

装订库是一个专用数据库，它由对象及其性质所组成。装订库本身具有安全性访问级别以限制访问服务器及其上有关安全性的信息。这个安全性被称为装订库访问级别(bindery access level)。一个访问级别确定谁能读取对象信息和谁能写或修改对象信息。装订库访问级别由表2—1所确定。

表2—1　装订库访问级别

4 = 装订库(Bindery)或OS访问
3 = 管理员(Supervisor)访问
2 = 对象(Object)访问
1 = 登录(Logged)访问

1.2.1　装订库对象

装订库API可用于创建对象，定义了它们的访问级别，以及定义这些对象的特征。对象特征被称为性质。装订库访问级别的性质能被加到对象上以限制访问所占有的资源以及从程序上证实被指定的性质值。一个对象的访问级别由读访问和写访问所组成。当使用装订库API时，读访问级别由低位半字节所定义(0xXR)，而写访问级别由高位半字节(0xWX)所定义。例如，一个对象的名字是LOGGED访问。在注册之前，该对象能被证实其存在，这是很必要的。写级别可保护一个性质不被错误的个人所改变。这就是LOGIN-CONTROL性质的成立(true)。它拥有写访问级别3。这允许仅管理员或OS改变注册限制。

读访问级别也可用于保护一个性质。例如，PASSWORD(口令)性质具有一个写访问级别2，它允许用户改变这个口令。然而，读级别是4，这要求口令通过OS已提供的专用API来改变，因为只有OS可读出装订库中包含的加密口令。当改变此口令时，API必须用和新口令一样的老口令通行。因此，当一个对象注册时，仅由知道老口令的某个人来改变口令。这就保护了用户口令免遭任何人(包括管理员)的改变。

管理和创建对象可以从程序上通过使用装订库 API 来实现。此外，装订库 API 允许应用关闭该装订库以让它进行归档并接着重打开它，禁止对个人或所有用户注册，并检测入侵者。

当创建一个对象时，一个应用必须给它指定一个类型和一个访问级别。装订库对象具有一个最多 48 个字符长的名字，包括零(空)结束符。它们也含有两字节的类型和一字节的访问级别(0x31)。装订库对象利用 CreateBinderyObject 调用来创建。当一个装订库对象启始被建立时，它并不具有与其相关的性质。为了成为一个所定义类型的合适对象，该对象必须拥有定义为那种对象类型的性质。对于用户的性质确定该对象已访问什么资源和文件服务器上的什么信息。这种访问可包含目录访问以及甚至限制注册到文件服务器(当用户被允许时)。一个用户对象需要拥有表 2—2 所列出的正确配置的性质。

一个用户(USER)对象也可拥有表 2—3 中列出的性质。这些性质是利用 CreateProperty 函数来建立的。这些性质的值是利用 WritePropertyValue 函数调用进行设置的。

表 2—2 USER 对象性质

GROUPS_I'M_IN	0x31 STATIC/SET
SECURITY_EQUALS	0x32 STATIC/SET
LOGIN_CONTROL	0x32 STATIC/ITEM
PASSWORD	0x24 STATIC/ITEM

表 2—3 其他的 USER 对象性质

IDENTIFICATION	0x31 STATIC/ITEM
OLD_PASSWORDS	0x24 STATIC/SET
NODE_CONTROL	0x32 STATIC/ITEM

为了拥有 OLD_PASSWORDS 性质，管理员必须要求该口令是唯一的且要求周期性改变。当 SYSCON 接收这个指向时，它此时创建了 OLD_PASSWORDS 性质，而文件服务器由此保持它。NODE_CONTROL 性质的建立是用于控制一个用户可从其进行注册的工作站。ACCOUNT_LOCKOUT 性质控制试图注册的次数。

装订库性质被分类为条目(items)或组合(sets)。一个条目可具有单一值，而组合可拥有多个值。一个性质由一个或多个 128 字节的段所组成。当一个值被改变时，此变化必须发生在相应的 128 字节段内。一个条目性质至少拥有单一段，而一个组合性质也可拥有一个或多个段。这些性质也具有像对象那样的装订库访问级别。读出性质段按顺序实现。当对于那个性质不存在多个段时，ReadPropertyValue 功能调用带有一个被设置的标记。

一个装订库对象可以是静态的或动态的。一个静态对象被写入磁盘，而即使服务器关闭时也是可用的。一个动态对象是一个包含在装订库中的临时对象，它位于内存中但未被写入磁盘，因而当文件服务器关闭时就会丢失它。网络上的服务器是动态对象，通过 NetWare Service Advertising Protocol(服务通告协议，SAP)可发现它并放置于装订库中。

1.2.2 装订库对象结构

装订库由服务器 SYS:\SYSTEM 目录中的几个文件所组成。这些文件含有装订库对象、

性质和这些性质的值。在这些文件中,入口的逻辑结构示於图 2—1 中。

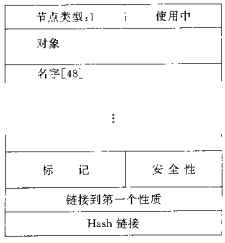

图 2-1 装订库对象结构

对象结构的节点类型(Node Type)域指出该结构是否被用于一个对象或某个其它的装订库构造。在使用中(In Use)域指出装订库的这个节点是否在使用中。该 ID 是 4 字节装订库对象 ID。类型(Type)域含有装订库对象类型之值。对于类型 USER,这个值就是 Ox0001。名字(Name)域含有 48 字节装订库对象名。标记(Flags)域描述该对象是否是静态的或动态的,一个组合还是一个条目。安全性(Security)域是该对象的装订库访问级别。链接到第一个性质(Link to First Property)是一个指针,它指向该对象所拥有的第一个性质结构的单元。若这个链接是零,那么该对象没有性质。Hash(散列)链接是对于装订数据库中下一个节点的链接。

其它的装订库对象包括表 2—4 中所找到的类型。

从 Novell 公司可获得一个装订库对象类型作为开发人员应用编程之用。这是由 Novell 公司的开发人员关系程序所管理的。

表 2—4 装订库对象类型

OT_WILD	OxFFFF
OT_USER	Ox0001
OT_USER_GROUP	Ox0002
OT_PRINT_QUEUE	Ox0003
OT_FILE_SERVER	Ox0004
OT_JOB_SERVER	Ox0005
OT_GATWAY	Ox0006
OT_PRINT_SERVER	Ox0007
OT_JOB_QUEUE	Ox000A
OT_ADMINISTRATION	Ox000B

OT_REMOTE_BRIDGE_SERVER	0x0024
OT_ADVERTISING_PRINT_SERVER	0x0047

2.2 用户对象和性质

用户对象的不同性质定义了用户是谁和内容。每个性质具有一个专门的功能。

2.2.1 GROUPS I'M IN

GROUPS I'M IN 性质定义一个用户属于什么组。GROUPS_I'M_IN 是一个 STATIC/SET 性质。它包含用户所归属的每个组的装订库对象 ID。这些组被指定受托者权限、安全性等效等。当一个用户是一个组的成员时，用户可继承该组所拥有的所有权限。这个性质的一个例子及其值见图 2—2。

GROUPS_I'M_IN	STATIC/SET	SUPERVISOR WRITE/LOGGED READ
	EVERYONE	

图 2—2 GROUPS_I'M_IN 性质打印

2.2.2 SECURITY EQUALS

SECURITY_EQUALS 性质定义对象是等同的而能操作同样权限的其它组或用户。这个性质是一个 STATIC/SET 性质。安全性等效仅被授于另一个对象，并且同样不允许用户对象继承另外对象的安全性等效。例如，若 JOE 在他的 SECURITY_EQUALS 性质中拥有管理员的对象 ID，他就拥有管理员的所有权限。然而，若 MARY 在她的 SECURITY_EQUALS 性质中拥有 JOE 的对象 ID，她仅继承 JOE 的个人权限，不是管理员权限。SECURITY_EQUALS 性质的打印结果列于图 2—3。

SECURITY EQUALS	STATIC/SET	SUPERVISOR WRITE/OBJECT READ
	EVERYONE	

图 2—3 SECURITY_EQUALS 性质打印

2.2.3 IDENTIFICATION

IDENTIFICATION 是一个 STATIC/ITEM 性质。它能被设置到管理员所希望的任意值，但打算描述指定用户帐户的个别用户。这个性质由单一的段组成。这个性质的打印结果示於图 2—4。

```
IDENTIFICATION STATIC/ITEM      SUPERVISOR WRITE/LOGGED READ
4c 6f 72 69 20 47 61 75 74 68 69 65 72 00 1c 98   Lori Gauthier...
1e 8b 1e 7b 1c 43 1c 11 1c fe 1b ed 44 10 4e 01   ...{.C......D.N.
16 f2 9d 0e 00 f2 5d 5d 5d 5d 78 77 3c 7c 20 7c   ......]]]]xw<| |
00 02 08 00 04 7c 5c 00 41 53 46 72 6d 7e 06 00   .....|\.DEFrm~..
5d 5d ea 7b 55 01 8a 52 00 02 3c 7c 5d 5d 08 00   ]]{U..R..<|]]..
20 7c 5d 5d 16 00 06 00 6d 7e 61 a2 61 o4 61 fa    |]]....m~a.a.a.
00 00 3c 7c 5d 5d 00 02 20 7c 5d 5d 2c c5 08 00   ..<|]]..|]],...
2c da 2c df 2c a4 31 f5 30 c5 30 8b 2f 31 2f f3   ,.,.,.1.0.0./1/.
```

图 2－4 IDENTIFICATION 性质打印

2.2.4 LOGIN_CONTROL

LOGIN_CONTROL 性质是一个 STATIC/ITEM 性质。LOGIN_CONTROL 性质用于保证文件服务器的安全，并限制在任意时刻对它的访问。LOGIN_CONTROL 性质由装订库段组成。利用 LOGIN_CONTROL 性质经由文件服务器的系统安全性检查发生在半小时内。若一个用户注册时间超过允许时间或者它的注册控制性质已改变，这允许文件服务器强迫该用户脱离系统。在这个段中的不同偏置代表不同的信息，而这些偏置值确定拥有该性质的对象的注册访问。在 LOGIN_CONTROL 性质中所含有的域显示於表 2－5 中。

只有最小口令长度(Minimum Password Length)、老口令(Old Passwords)、注册次数(Login Times)、以及允许的注册站点限制(Allowed Login Stations restrictions)影响管理员。

表 2－5 LOGIN_CONTROL 性质

偏置	域	大小
0	Account Expiration Data	3 字节
3	Account Disabled Flag	1 字节
4	Password Expiration Data	3 字节
7	Grace Logins Remaining	1 字节
8	Password Expiration Interval	2 字节
10	Grace Login Reset Value	1 字节
11	Minimum Password Length	1 字节
12	Maximum Concurrent Connections	2 字节
14	Allowed Login Time Bitmap	42 字节
56	Last Login Data And Time	6 字节
62	Restriction Flags	1 字节
63	Unused	1 字节
64	Maximum Disk Usage in Blocks	4 字节
68	Bad Login Count	2 字节

偏置	域	大小
70	Next Reset Time	4 字节
74	Bad Login Address	12 字节

Account Expiration Data 域表示帐户截止日期。第一字节是年，第二字节代表月，而最后字节是该月的日期。当这每一个字节被设置为 0 时，帐户将不会截止。

Account Disabled Flag 是一个布尔标记，它表示该对象是否已被禁止(00 = 许可，FFh = 禁止)。这个域允许管理员在不需要消除一个用户的情况下禁止对系统的访问。这特别适用于管理客人帐户或用户并非频繁访问其帐户的帐户。这个标记由系统每隔半小时就进行检查。若用户注册时这个标记已被设置，那么一个报文被送往用户进行注销。在 5 分钟之后，该连接被消除，而所有注册将接收一个错误(0xDC)，它表示帐户被禁止。这个标记并不对管理员用户产生任何影响。

Password Expiration Data 指出一个口令何时将截止(期满)。这个域是三个字节长，而该日期用 Account Expiration Field 相同的方式被包含。当口令用下述手段改变时，这个值被重新设置。其手段是将 Password Expiration Interval 中包含的日期数附加到口令被改变的日期——除了本节后面要讨论的特定情况之外：当拥有管理员权限的某个人改变口令时，该日期被设置到 Jan 1，1985 以强迫用户重新指定它自己的口令。

当一个用户用一个已截止的口令访问服务器时，就发生 Grace Login。Grace Logins 的控制范围可从零到许多。Grace Logins Remaining 指出一个用户在没有改变其口令情况下能注册的次数。当一个用户改变其口令时，这个值由 Grace Login Reset Value 重新设置，或者它能由管理员人工地重设置。当一个系统需要十分安全，Grace Login Values 应很低的。

Minimum Password Length 域允许管理员通过增加一个口令必须有的最小长度来减少偶然的或中断系统访问的风险。安全性研究证实，当一个口令是 3 字节或更少，安全性就易于被破坏。

Maximum Concurrent Connections 域允许控制一个用户可同时拥有的连接数。这个功能允许管理员将用户的连接数控制到一个办公室中的工作站数。这也减少了一个用户将保留一台不参与的注册工作站的风险。

Allowed Login Time Bitmap 是一个 42 字节的数组，其中每个位是等于 $\frac{1}{2}$ 小时间隔。整个数组表示在一周内的总时数的一半。当一个对象可注册时，这个域允许网络管理人员拥有限制次数的能力。当该位有效时，用户可在规定的时间内访问服务器。当备份并不发生时以及当一个用户由于下列原因不应工作时，即无论是需要用户不保留其注册工作站的安全系统还是避免超时工作，使用这种功能性将允许管理员限制注册。

Last Login And Time 含有对象的最近注册日期和时间。这个域由年、月、日、小时、分、秒组成，每次成功的注册到文件服务器都会产生这个域的一次更新。

Restriction Flags 含有 8 位，它控制 PASSWORD 和 OLD_PASSWORDS 性质。若位 0 被设置(0x01)，那么仅仅管理员可改变此口令。在此情况下，Password Expiration Date 并未被设置到 January 1，1985。若位 1 被设置(0x02)，那么 OLD PASSWORDS 性质是存在的，并需要进行更新。当一个口令变化时，可以检查此位以确定该口令是否必须是唯一的。这一位是可用

的，因为一个用户并不必须访问 OLD_PASSWORDS 性质，即使知道它的存在。

Maximum Disk Usage In Blocks 域是一个用户磁盘空间被限定的服务器磁盘块数。未限定的磁盘空间由 0x7FFFFFFF 所表示。限定磁盘空间是必须在安装期间被许可的选项。在 NetWare 2.x 中的磁盘块是 4K 大小，2.x 中的磁盘块限制是文件服务器范围的，或者是所有卷上的所有磁盘块之和。NetWare 3.x 中的磁盘块可配置于安装过程。3.x 上的磁盘块限定是以每卷为基础的。这允许一个网络管理人员限制重要的卷上的磁盘空间，而并非在不需要限制的卷上。若包括管理员在内的用户在限制达到之后试图写入到磁盘，工作站将接收到一个 DISK FULL 出错信息。属于 NetWare 支持的应用应理解这个错误，并向用户指出该错误是物理的还是虚拟的。

不成功的注册信息和成功的注册信息一样由 LOGIN_CONTROL 性质所保持。用 LOGIN_CONTROL 性质描述入侵者检测的域包括 Bad Login Count、Next Reset Time 和 Bad Login Address。仅当文件服务器上的对象拥有 ACCT_LOCKOUT 性质时，这些域才被保护。当那个性质可用时，就跟踪入侵者检测信息。

Bad Login Count 是该值被重新设置以来非法注册的计数。当发生一个成功的注册时或者当来自 ACCT_LOCKOUT 性质的重设置分钟数期满时，这个值被清除为 0。若一个帐户已被禁止，这个域被设置为 0xFFFF。

Next Reset Time 是自 Jan 1，1985 年开始的秒数，它表示 Bad Login Count 将被重新设置的时间。

Bad Login Address 是注册失败站点的 12 字节网际地址。这个域许可网络管理人员跟踪入侵者试图注册的地点。

2.2.5 PASSWORD

PASSWORD 性质是一个 STATIC/ITEM 性质。它具有装订库访问级别 0x24。从 NetWare 2.15 开始，口令在线上被加密。所使用的加密算法是非可逆的。该口令此时以加密形式被存储在装订库中。当由用户给出的一个口令同 PASSWORD 性质中所包含的值相比较时，它是用加密形式来实现的。若这些密码并不匹配，那么口令被拒绝。

2.2.6 OLD_PASSWORDS

OLD_PASSWORDS 性质是一个 STATIC/ITEM 性质。它包含对象已使用的最近 8 个口令和加密格式的用户名字。这个性质用于限制一个口令如何能经常使用。当用户对象改变口令时，这个性质允许文件服务器存储先前口令的加密形式。对象名字也存储在这里，这是为了不允许对象使用它自己的名字作为口令而增加安全性。

2.2.7 NODE_CONTRGL

NODE_CONTROL 性质是一个 STATIC/ITEM 性质。这个性质确定一个用户可从哪个节点进行注册。从任何别的节点进行注册将促使用户被拒绝。该性质值由网络和用户注册的允许位置的节点所组成。网络地址不能被设置为 00000000 或 FFFFFFFFh。若该节点被设置为 FFFFFFFFFFFFh，那么该用户能从指定网络上的任意节点进行注册。