第一章　Win32 操作系统

当 Windows NT 设计组开始创造 Win32 操作系统时，他们的头脑中有一套明确的工程目标。他们完成了其中大多数目标，但在其它一些目标上故意接受了折衷方案。

他们的努力成果是一种新型操作系统。这种操作系统不同于先前设计的任何 Windows 操作系统，其功能借签了早期的 Windows 系统、基于正文的 MS—DOS 操作系统、UNIX 之类的客户/服务器多用户系统，甚至多处理器系统，如 NCR 制造的 multi—80486 系统。

Windows NT 的新功能很多，而这些新功能又来自众多不同的出处，因此在编程方面向 Windows 软件的开发者提出了独特的挑战。

在 Microsoft 发布的 Windows 系统中，Windows NT 第一个具备平板式 32 位内存映象结构。第一个支持多线程，第一个支持客户/服务器网络，第一个具备内部安全功能，以及第一个在设计意图上支持开放系统。

1.1　Windows NT 的新功能

Windows NT 的新功能可分为四大类：

* **支持开放系统**。Windows NT 支持网络通信 —— 不只是 Windows for Workgroups 和 Microsoft LAN Manager 所提供的那种端对端通信方式，而是一种真正的信息高速公路式网络，支持使用客户/服务器通信方式和流行的通信协议，如 TCP/IP。
* **支持多线程**。Windows NT 能够为各个应用程序分配不同的 CPU 时间片，同时运行多个程序。这种抢占式多任务功能是由线程管理技术实现的。一次管理一个以上的线程就叫多线程。Win32 操作系统根据预先确定的任务优先级为正在处理的每个任务分配一部分处理器时间，从而自动地管理多线程。
* **网络支持**。Windows NT 是第一个支持客户/服务器网络的 Windows 系统，并且支持分布式计算方式。Windows NT 能通过客户/服务器模型把计算任务分派到网络中的多台计算机上，使每台工作站比单台机器提供更大的计算能力。
* **安全性和可靠性**。Windows NT 的安全功能已达到美国政府认可的 C2 安全级别。其登录系统既保护各个用户的数据，也保护系统管理员的数据。系统的设计意图还包括保护内存数据的完整性。存储在内存中的每个 32 位应用程序具有专用的地址空间，不能修改其地址空间之外的内存。这意味着，应用程序不能破坏其它应用程序或操作系统所控制的内存区域。

本章的剩余部分将更详细地阐述所有这些功能。在随后几章里，读者将学会如何把这些功能用到自己的 Windows NT 程序中。

1.2 Win32 操作系统模型

Windows NT 操作系统是两种传统操作系统模型的混合体：一种是层次模型，另一种是客户/服务器模型。

在层次型操作系统中，构成操作系统的各个模块是相互分开的，它们根据功能关系组合在一起并分层排列。任一层的模块只能调用该层之下的代码，层次型操作系统如图 1.1 所示。

1.2.1 层次型操作系统

在图 1.1 所示的层次型操作系统中，构成操作系统代码的各个过程不能随意地相互调用。每个模块都含有一组可供其它模块调用的函数，但任一层中的代码只能调用较低层中的代码。

当应用程序加入这一堆代码时，它所能调用的代码只能位于图 1.1 中的最高层：系统服务层。这一限制措施能防止应用程序存取用户编写的应用程序接触敏感的操作系统代码。

"面条式"程序设计

散漫的非结构化源代码难以理解，通常被称作"spaghetti（一种意大利面条）代码"，作为对照，一些专家认为，层次型操作系统是用"lasagna（另一种意大利长扁面条）代码"写成的。

在层次型操作系统中，每个代码模块只能存取下层代码，所以在总体上保护了系统免受各个代码模块可能产生的危险作用。层次型操作系统的另一个好处是，模块化设计方法使系统尤其易于升级和维护。最后，层次型操作系统比其它一些类型的系统更容易调试。堆栈底层的代码可以先调试，然后依次调试其它层次的代码，直到整个系统无误运行为止。

1.2.2 客户/服务器操作系统

一些现代操作系统没有按层次模型进行设计，而是按客户/服务器模型进行设计的。在客户/服务器操作系统中，任务被划分为多个进程。每个进程实现一组服务，如内存服务、文件服务和通信服务。与每种服务相关连的有运行在用户状态下的服务器模块和运行在系统中的任何应用程序。

Windows NT 操作系统的客户/服务器结构不应该与客户/服务器网络混淆，后者在概念上与前者类似，但实现方式不同。客户/服务器网络在第七章"网络"、第八章"管道"和第九章"Windows NT Sockets"中描述。

1.2.2.1 内核和微核

在根据客户/服务器模型构造的典型操作系统中，只有一个模块运行在内核状态中，这个模块叫微核。当应用程序需要得到一项服务，如内存管理服务时，就向微核发送一条消息，申请这项服务。微核把消息传递给适当的服务器，由服务器执行此操作，操作完成时再通知微核。微核随后向应用程序送回一条消息，把操作结果通知应用程序。

使用客户/服务器操作系统的主要优点是系统的每个部件都小巧玲珑自成体系。每个服

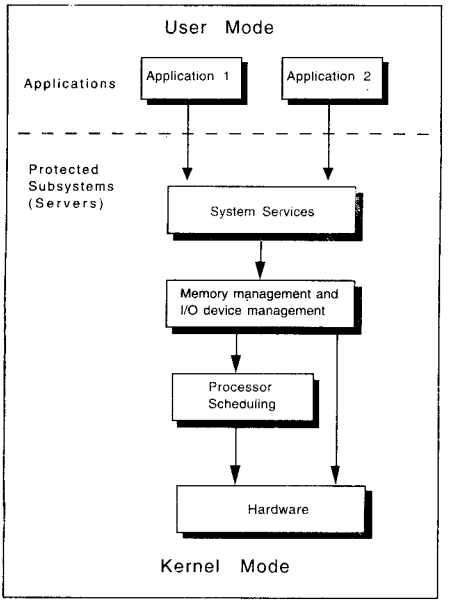

图 1.1 层次型操作系统

务器在不同的用户态进程中运行(参见本章后面的"进程和线程"),因此,服务器的失败甚至重新启动 —有可能不会使系统崩溃或破坏服务器的任何数据。另一个优点是,不同的服务器可以运行在(多处理器计算机的)不同微处理器上,甚至运行在网络中的不同计算机上。

1.2.2.2 客户/服务器系统的各种类型

客户/服务器操作系统有许多类型。在某些系统中,微核执行多种任务。在另一些系统中,微核只执行少量任务。

图 1.2 是客户/服务器操作系统的简图。其中微核只是用一个起传递消息作用的模块表示。在实际应用时，微核通常还执行其它任务，如调度线程、管理虚拟内存和控制设备驱动程序等。

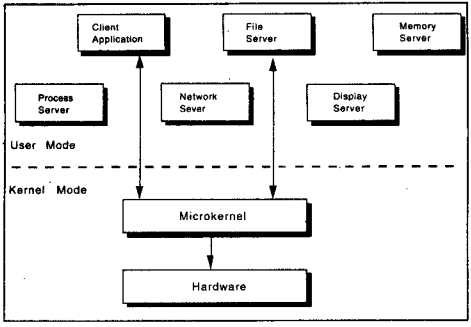

图 1.2 一种客户/服务器操作系统

1.2.3 Windows NT 操作系统模型

Windows NT 操作系统具有层次型操作系统模型和客户/服务器模型两者的特征，其部分结构如图 1.3 所示。总的来说，Windows NT 操作系统工作起来更象一个客户/服务器系统而不象层次系统。但 NT 系统的内核状态部分含有层次系统的一些成分，这些成分包括内核状态部分的低层部分，这一部分包括内核本身和一个名叫硬件抽象层(HAL)的模块。

用户状态和内核状态

注意，在图 1.3 中，一条粗线把应用程序层与下面各层分开。这条线上面的应用程序层被标记为"用户状态"，这条线下面是操作系统模块出现的地方，它们被标记为"内核状态"。

这种划分反映了 Windows NT 操作系统与大多数操作系统一样，把正在执行的代码划分成两种不同的类别或状态。Windows NT 在优先的进程状态中运行自己的系统代码，这些代码有权使用计算机硬件和系统数据。应用程序在非优先状态中运行。

构成 Windows NT 内核状态部分的大多数组件都组合在一个更大的部件，即 NT 执行部件中。NT 执行部件的各个部分执行下面这些系统任务：虚拟内存管理、资源管理、输入输

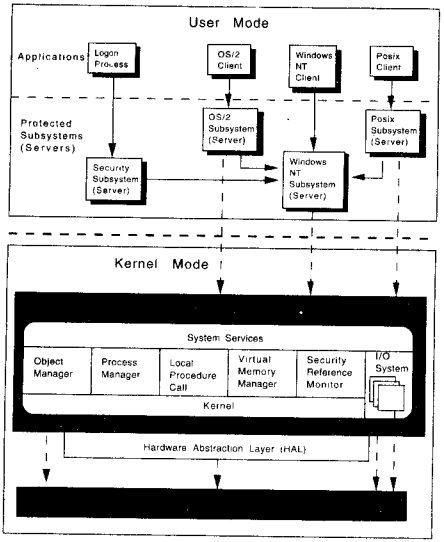

图 1.3 Windows NT 操作系统

出(I/O)操作文件管理,以及对网络驱动器、进程之间的通信和 Windows NT 安全系统部分的管理。此护在NT执行部件中的各个组件相互间通过一套精心编制的内部例程发生作用。

刚好位于 HAL(硬件抽象层)之上的 Windows NT 内核执行低层的操作系统功能,这些功能类似于客户/服务器操作系统的微核所完成的那些功能,如调度线程、分派中断异常以及同步多个处理器。NT 内核还包括一套例程和资源,以便执行部件的其它组件,用它们来实现高级任务。

内核下面是 HAL,这些软件把内核和 NT 执行部件的其余部件同运行 Windows NT 的硬件平台隔开。在能够运行 NT 兼容型操作系统的硬件平台上,HAL 是为了与硬件直接通信而专门设计的。因此,为了使 Windows NT 与多种硬件系统保持兼容性,HAL 部分担负主要责任

出于偶然因素，HAL 的实现方式是动态链接库(DLL)。DLL 是当例程在运行期间发生调用时赋给这些例程的库函数。DLL 是第四章的主题。

Windows NT 的用户状态模块将在下一节详加解释。有关内核状态模块更详细的说明，参见本章后面的"内核状态"一节。

隐藏的代码

Windows NT 的设计者尽量使系统易于移植，并限制依赖于特定硬件体系的代码数量。正如完成情况表明的那样，面向处理器的代码是必需的。于是，NT 设计组把这类代码放在 NT 内核的最底层。这一小部分面向处理器的代码因此便与操作系统的其余部分分隔开了。

设备驱动程序当然包含面向设备的代码，但这些代码调用 NT 内核例程和 HAL 例程，避免了对处理器和平台的依赖。

依赖于平台的代码，即依赖于某个厂家的计算机系统(如 MIPS R4000)的代码，位于硬件抽象层(HAL)，并由第三方厂家提供。

1.2.4 用户状态模块

NT 操作系统的用户状态模块含用多个组件，它们执行通常与操作系统有关的大多数任务。Windows NT 应用程序作为客户运行在用户状态中。应用程序编程接口(API)也在用户状态中运行，但它们是服务器。这些服务器有时叫作受保护的子系统。图 1.4 说明了这种安排。

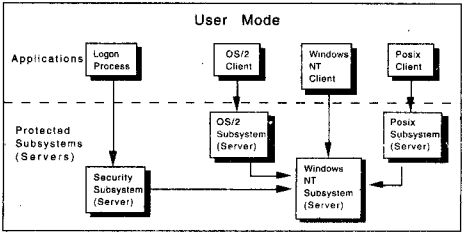

图 1.4 Windows NT 的用户状态模块

应用程序编程接口(API)

API 是一套库函数或接口程序，它们使应用程序通过自身的编程语言获取低层模块提供的功能，如操作系统、图形用户界面(GUI)和通信协议。应用程序有多种类型，因此，API 也有多种类型。

在图 1.3 中，标记为"系统服务"的代码层可以称作 API。从更广的范围看，

Win32 系统本身，即应用程序和 Windows NT 操作系统之间的整个接口部分，也是 API。实际上，Win32 系统通常也叫做"Win32 API"(Win32 系统是长达五卷的《Microsoft Win32 程序员参考手册》的主题)。

图 1.4 下半部分的服务器被标记为受保护的子系统。这是因为每个服务器在不同的进程中运行，每个进程使用的内存不同于其它进程使用的内存。这种分隔措施能保护每个进程使用的内存空间，使之免受其它进程活动的侵犯行为所带来的潜在危害。

由于用户状态模块中受保护的子系统并不自动地共享内存，它们通过传递消息进行通信，如图 1.5 所示。图 1.5 中的实线代表消息在客户和服务器之间以及在各个服务器之间采取的路径。

受保护的子系统所分派的所有消息都穿过 NT 执行部件，如图中带箭头的虚线所示。NT 执行部件是这种操作系统的引擎，它能够支持任何数量的服务器进程。服务器是 NT 执行部件的用户，也是编程接口，还为不同类型的应用程序提供执行环境。

每个受保护的子系统作为服务器，所起到的作用是提供可供程序调用的 API。当应用程序或其它服务器需要调用 API 例程时，一条消息便送到该 API 例程所在的服务器上。这条消息是经由 NT 执行部件的本地过程调用(LPC)机构发送的。这种消息传递机构驻留在操作系统的系统服务模块中。服务器执行完 API 例程后，通过发送另一条 LPC，把结果送回应用程序进程。图 1.5 说明了这个过程。

本地和远程过程调用

NT 服务器与执行部件进行通信时使用的 LPC(本地过程调用)系统是根据 RPC(远程过程调用)改编的。在分布式计算系统上，当工作站需要存取服务器提供的函数时，便使用 RPC。

远程过程调用技术是一种经典方法，用于调用网络中远端机器上的代码。某台机器上的调用程序把函数所需的参数包裹在信报中，然后通过网络发往另一台机器。这台机器上的远端过程对信报做相反的处理，即解开调用参数，执行函数，并把回件发给调用程序所在的机器。

1.2.4.1 Win32 子系统

Win32 API 包括用于 POSIX、OS/2、16 位 Windows 和 MS—DOS 环境的子系统。但最重要的子系统是 Win32 子系统，这一超级服务器使应用程序能够使用 32 位的 Windows API。

Win32 环境子系统提供 Windows NT 的图形用户界面(GUI)并控制用户的所有输入和应用程序的所有输出。其它子系统通过 Win32 子系统接受用户输入和显示输出。这些子系统可以按需要装入内存，与其它子系统一同运行。

1.2.4.2 环境子系统与集成子系统

Windows NT 有两种受保护的子系统：环境子系统和集成子系统。环境子系统是运行在用户状态下的服务器，它提供面向某个操作系统的 API。集成子系统是在操作系统和一些重要的设施(如网络服务)之间提供接口的服务器。

环境子系统使 Windows NT 有能力运行某个操作系统环境，如 Win32、16 位 Windows、DOS、POSIX 和 OS/2。Windows NT 同时运行这些环境子系统。因此，Windows NT 系统能

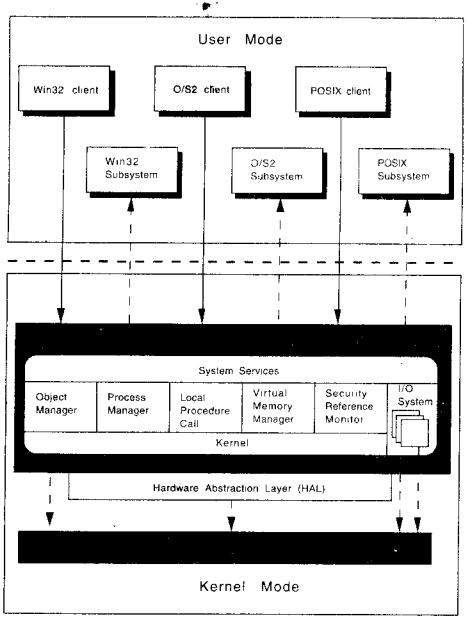

图 1.5 Windows NT 操作系统中的客户与服务器

够运行为其它环境，如 16 位 Windows 和 MS-DOS 编写的应用程序，同时还运行专门为 Win32 API 编写的应用程序。

1.2.4.3 Win32 环境子系统

Win32 环境子系统提供 Windows NT 的用户接口，它控制视频显示、键盘、鼠标器和连接在本地计算机上的其它输入设备。它还是 Win32 应用程序的服务器，能够提供 Win32 API。

Win32 子系统不控制非 Win32 应用程序的执行，也不管理 16 位 Windows 应用程序、

DOS 应用程序或 POSIX 应用程序。用户运行应用程序时，如果 Win32 子系统识别出这个应用程序不是 32 位 Windows 程序，就先判明这个程序的类型，然后调用其它子系统运行此程序。

每个环境子系统都提供一套 API 供相应的客户应用程序使用。例如，Win32 系统提供 32 位 Windows API 例程，而 OS/2 子系统提供 OS/2 API 例程。

·运行 DOS 应用程序和 16 位 Windows 程序

Windows NT 靠模拟 DOS 和 Windows 3.X 环境来运行 MS－DOS 应用程序和 Windows 3.X 应用程序。Windows NT 用名叫 DOS 虚拟(VDM)机的环境子系统来完成模拟工作。Windows NT VDM 能模拟整个 MS－DOS 环境。

在 Windows NT 中运行的每个 MS－DOS 应用程序或 16 位 Windows 应用程序分属不同的 VDM 进程范围。因此，多个 VDM 进程可以同时运行，这是 VDM 进程的独特之处。其它类型的环境子系统进程一次只能有一个在运行(有关 VDM 进程的详细说明，参见本章后面的"DOS 虚拟机"一节)。

用户所见到的全部就是 Windows NT

Windows NT 环境子系统能支持许多客户应用程序。每个子系统记录各个客户的信息并维护所有客户应用程序所共享的任何全局信息。尽管 Windows NT 能够在任意时候启动若干子系统和 VDM，但唯一让用户见到的环境子系统只有 Win32 子系统，对用户来说，似乎 Windows 正在运行所有的应用程序。

·管理视频输出

所有的视频输出由 Win32 子系统处理。因此，其它环境子系统必须把应用程序的视频输出结果定向到 Win32 子系统，输出结果才能显示出来。正在运行 16 位 Windows 应用程序的 VDM 把应用程序的输出调用转换成 Win32 调用，然后向 Win32 子系统发送一条请求显示的消息。

OS/2 和 POSIX 子系统以及运行 MS－DOS 应用程序的 VDM 都把应用程序的字符型输出结果定向到 Win32 子系统。Win32 子系统在字符型控制台窗口显示输出结果。控制台窗口是第五章我们要讨论的内容。

1.2.4.4 安全子系统

除了 Win32 子系统和其它 NT 环境子系统外，用户状态模块还包含其它一些受保护的子系统，它们属于集成子系统。这些集成子系统完成各种不同的操作系统功能，特别值得一提的一个集成子系统就是安全子系统。

Windows NT 环境子系统有时被称作受保护的子系统，这是因为它们被相互隔离，也就互不干扰。其中一个受保护的子系统，即安全子系统，负责管理本地计算机上的安全保护机制，安全子系统记录哪些用户享有特权，哪些系统资源需要统计，以便用户存取。当应用程序访问受保护的子系统时，安全子系统还确定是否产生统计警报或统计消息。另外，安全子系统维护一个有关用户帐户的信息数据库，包括用户名、口令、用户组名(出于安全目的)以及用户享有的任何特权。安全子系统还接受用户的登录信息并起动登录防伪过程。

1.2.4.5 与网络有关的服务器

与网络有关的几个软件部分属于集成子系统。在最重要的网络类集成子系统(通常叫作

服务)中,有两个是工作站服务和服务器服务。这些服务的每一个都是在用户状态下运行的进程,分别实现一套 API 并管理一对与网络有关的实用程序;网络重定向器和网络服务器。图 1.7 是网络重定向器和网络服务器的示意图。

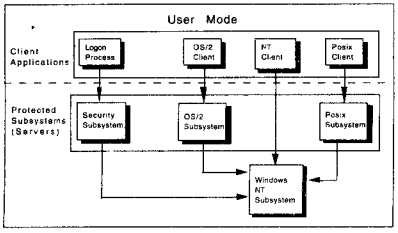

图 1.6 安全子系统

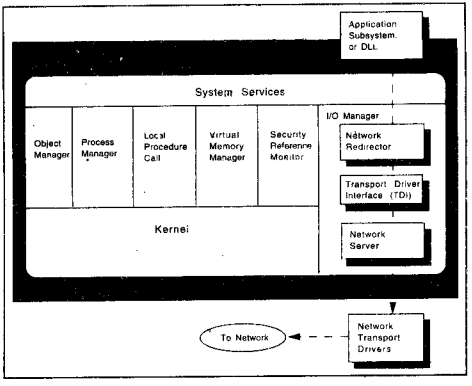

图 1.7 网络重定向器和网络服务器

当被存取的文件或设备不在本地，而是通过网络与本地机器连接在一起时，网络重定向器把 I/O 请求发送（重定向）到网络中。网络重定向器的请求由远端机器上的服务器接收。

网络重定向器和网络服务器在实现方式上都属于文件系统的驱动程序——也就是说，它们是 NT I/O 系统的一部分。有关这些内容更多的说明，请参见本章后面的“I/O 系统”一节。

1.2.5 在多处理器系统上使用 Windows NT

如同本章后面的“过程和线程”一节解释的那样，Windows NT 利用名叫线程的编程实体在进程之间分派 CPU 时间，使系统有可能根据优先级同时执行多个应用程序。

在单处理器计算机系统上，由于使用了多个线程（即多线程），这些线程似乎是在同时运行的。在单处理器系统上，这只是一种假象。实际上，这些线程是在系统分配的各个单独的 CPU 时间片中顺序执行各自的任务的。

当 Windows NT 在多处理器计算机，如 NCR 制造的 multi－80486 系统上运行时，Windows NT 没有必要表现出它正在同时执行多个线程的假象。当 Win32 系统在多处理器计算机上运行多个应用程序或任务时，它实际上真正地在同一时间执行多个线程，每个线程占用一个 CPU。

1.2.5.1 对称和非对称处理

具有多进程特征的操作系统分属两种类别：一些支持对称多处理（SMP），而另一些支持非对称处理（ASMP）。当一个处理器操作系统使用非对称处理时，它通过在一个处理器上运行操作系统代码，其它处理器运行应用程序的方式，将操作系统和用户编写的代码隔开。

1.2.5.2 非对称多处理

由于非对称多处理方式的操作系统只在一台处理器上运行 OS 代码，因此，开发 ASMP 操作系统并不特别困难，在现有的单处理器操作系统上做相对较少的改进就可完成所有工作了。

在非对称的硬件平台上，如一个处理器附加一个协处理器，或两个不共享所有可用内存的处理器，ASMP 操作系统工作得很好。问题是 ASMP 操作系统难以移植，不同厂家的硬件（甚至同一厂家的硬件的不同版本）通常具有不同的规范和非对称度。要么硬件厂家针对指定的操作系统制造硬件，要么大量改写操作系统，以适应硬件平台。

图 1.8 说明了 ASMP 操作系统的操作过程。

1.2.5.3 对称处理

与 ASMP 系统不同的是，SMP（对称处理）系统允许操作在任意的自由处理器上运行——甚至同时在所有处理器上运行。当 Windows NT 在多处理器计算机上运行时，它按照 SMP 系统的方式运行。

在多处理器系统中，SMP 方式（如图 1.9 所示）的效率更高，因为操作系统并不局限于只使用一个处理器。当操作系统只在一台处理器上运行时，系统可能使这台处理器的负担到达极限，以致慢得如同蜗牛爬行，而其它处理器却无所事事。

这种额外工作降低了整个系统的效率。与此相反的是，在 SMP 系统中，当操作系统需要额外的 CPU 时间时，它能够从空闲的处理器中获得。

除了提高效率之外，SMP 系统还能减少系统的停机时间，因为若有一个处理器发生故

障，操作系统代码可以切换到其它处理器上。

SMP 方式还有一个优点，由于不同厂家采用类似的方法制造对称型硬件，可移植的 SMP 操作系统更容易开发出来。

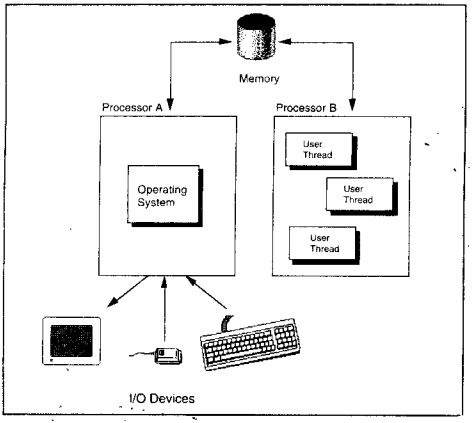

图 1.8 非对称多处理系统

1.2.6 内核状态

图 1.10 示意了如何把 NT 操作系统的内核状态部分分解成多个模块，这些模块包括文件系统模块、高速缓存管理程序、设备驱动程序和网络驱动程序。

1.2.6.1 I/O 系统

图 1.10 列出了一组负责处理来自各种设备的输入数据和向这些设备送回输出结果的子部件，这些子部件是：

* I/O 管理程序：实现与设备无关的 I/O 机构并为 NT 执行部件的输入/输出建立一个模型；
* 文件系统：文件系统由 Windows NT 驱动程序管理。这些驱动程序接受面向文件的输入/输出请求，并把这些请求转换成与某个设备相关连的 I/O 请求；
* 高速缓冲管理程序：在系统内存中存储最近的读盘信息，这种能力可提高基于文件的

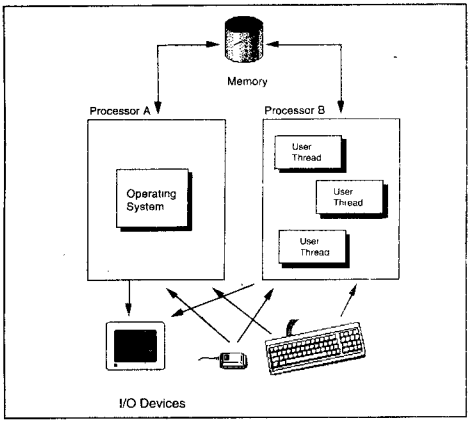

图 1.9　对称多处理系统

I/O 的性能。高速缓冲管理程序利用虚拟内存管理程序(VMM)的分页机制在后台自动把修改过的数据写回磁盘；

* 设备驱动程序，这些低级驱动程序直接操纵硬件，向物理设备或网络写回给出结果，或从这些设备中获取输入数据；
* 网络重定向器和网络服务器，它们都是文件系统驱动程序，前者向网络中的机器发送远端 I/O 请求，后者接收这种请求。

1.2.6.2　NT 执行部件

内核状态模块的心脏是 NT 执行部件。尽管 NT 执行部件有一个用户接口，但其本身仍是一个完全的操作系统。它由若干组件构成，每个组件实现两套功能。其中一套叫系统服务，可以被环境子系统和执行部件的其它组件调用。另一套由内部例程构成，只能供执行部件内的组件使用。NT 执行部件的各个组件如图 1.11 所示。

尽管 NT 执行部件提供的系统服务工作起来象 API，但执行部件在本质上不同于环境子系统。例如，执行部件不在自己的一个进程中连续运行。相反，它在现有进程的环境中运行，当出现重要的系统事件时，接管正在执行的线程。例如，当一个线程调用系统服务并被处理器陷入时，或者当外部设备中断处理器时，NT 内核就接着当前正在运行的线程的控制

权。NT 内核调用适当的系统代码来处理相应事件，执行代码后，把控制权交还给中断发生前正在执行的代码。

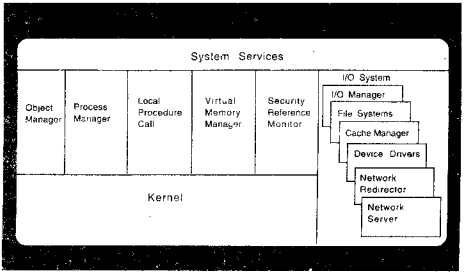

图 1.10 I/O 系统的各个部分

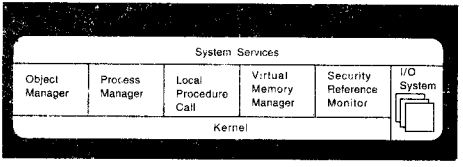

图 1.11 NT 执行部件的各个组件

NT 执行部件的各个组件是相互隔离的，并维护相互间的独立性，分别建立和操纵各自所需的系统数据。由于组件之间的接口得到了仔细的控制，故有可能从操作系统中完全移走一个组件，换上一个操作方式不同的组件，只要新的组件能正确地实现所有的系统服务和内部接口，操作系统便同以前一样运行。由于 NT 执行部件的各个组件按照可以预测的方式相互作用，操作系统的维护任务也更为容易了。

表 1.1 列出并说明了 NT 执行部件的各个组件。

1.2.6.3 NT 本机服务

NT 执行部件的各个组件所提供的服务叫 NT 本机服务。Windows NT 环境子系统靠调用 NT 本机服务来实现各自的 API。例如，虚拟管理器（参见“虚拟内存”一节）为环境子系统

提供内存分配和重分配服务。类似地，进程管理器提供用于创建和终止进程和线程的服务。

表 1.1 NT 执行部件的各个组件

组件	任务
内核	调度线程的运行，响应中断和异常，同步多个处理器的活动并提供一套基本对象和接口，供 NT 执行部件的其余部分用来实现高级对象。
对象管理器	创建、管理和删除 NT 执行部件的对象(用来表示操作系统资源的抽象数据类型)。
进程管理器	创建和终止进程和线程。进程管理器还暂停和重新执行线程，存储和检索关于 NT 进程和线程的信息。
I/O 系统	这一组组件负责处理来自各种设备的输入数据和向这些设备送回输出结果。
硬件抽象层(HAL)	这是一个动态链接库，提供 NT 执行部件和运行 Windows NT 的硬件平台之间的接口。HAL 隐藏了与硬件相关的细节，如 I/O 接口、中断控制器和多处理器通信机制。为了尽量维护兼容性，当 NT 执行部件的各个组件需要有关平台的信息时，调用 HAL 例程而不直接存取平台硬件。
本地过程调用(LPC)机制	在同一台计算机的客户进程和服务器进程之间传递消息。LPC 是远程过程调用(RPC)的一个灵活的优化版本。RPC 是客户和服务器在网络上进行通信的工业标准机制。
虚拟内存管理器	实现虚拟内存，这种内存管理系统为每个进程提供一个很大的专用地址空间并保护每个进程的地址空间不受其它进程的影响。当内存的使用率过高时，VMM 挑选出一些内存数据并把它们转移到磁盘上，再次使用这些数据时，把它们重新读回内存——这种做法叫做换页。
安全性咨询监视器	加强本地计算机上的安全性，保护操作系统资源，在运行期间保护和审计对象。

NT 本地服务是低级服务，主要供受保护的子系统、DLL 和 NT 执行部件的组件使用。在 Windows NT 上运行的应用程序通常不使用 NT 本机服务，而是调用 Win32 API 或 MS-DOS、16 位 Windows、POSIX 或 OS/2 等环境子系统提供的 API。

1.3 过程和线程

在 Windows NT 中，进程是应用程序的一个运行实例。进程大致等价于早于 NT 的 Windows 版本中的任务，它包含来自可执行文件的代码、全局变量和静态变量。进程可以拥有资源，如文件、动态内存分配块和线程。这些资源在进程的生命周期内创建，当进程终止时被销毁。

线程可以定义为应用程序中的一个执行单位。与每个线程相关联的有一个 CPU 指令序列、一组 CPU 寄存器和一个栈。在 Windows 3.X 应用程序中，一个任务只有一个线程，这个线程通常由应用程序的 WinMain 函数中的事件循环来控制。因此，Windows 3.X 应用程序只有一条代码路径。

在 Windows NT 中，一个进程可以拥有多个线程，线程可以创建其它线程。Windows

NT 内核用名叫调度器的服务为正在运行的每个线程分配 CPU 时间。在多处理器计算系统中，各个线程可以运行在任一台处理器上。

创建一个 Win32 进程后，系统自动地为这个进程创建一个线程，这个线程叫进程的主线程。主线程能创建其它线程，而这些线程都能够创建更多的线程。

1.3.1 虚拟内存

Windows NT 带有一个磁盘高速缓冲系统，这个系统能够自动地在内存和硬盘之间来回移动数据，它为应用程序提供的虚拟内存比实际存在的物理内存数量多得多。虚拟内存由名叫虚拟内存管理器(VMM)的系统服务管理。

虚拟内存管理器通过缓冲磁盘文件的读写为正在运行的每个进程提供 4GB 虚拟内存地址空间的存取能力，这个虚拟内存空间中的 2GB 空间供得到这一虚拟内存的进程使用，另外 2GB 由系统使用。在当今的技术范围内，或者在可以预见的将来，这种策略所提供的内存比应用程序可能需要的内存多得多。

图 1.12 说明了虚拟内存管理器如何实现虚拟内存。在图中，有两个 Windows NT 进程正运行在一个拥有 32MB 物理内存的计算机上。虚拟内存管理器已经为每个进程分配了 4GB 的虚拟地址空间。因此，每个进程都觉得有 4GB 的内存可供使用。

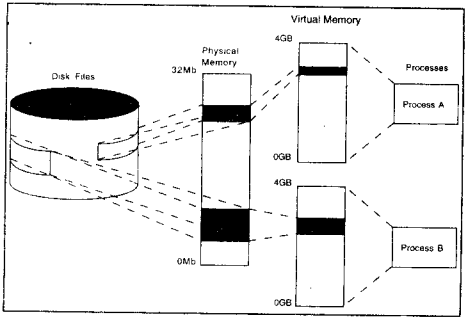

图 1.12 Win32 如何管理虚拟内存

1.3.1.1 进程如何使用虚拟内存

Windows NT 为驻留在 RAM 中的每个进程分配了 4GB 的虚拟内存空间，在这个空间中，Win32 操作系统保留上端的 2GB 供自己使用，虚拟内存下端的 2GB 则分配给进程。

图 1.13 说明了 Windows NT 如何为进程分配虚拟内存以及如何划分分配给每个进程

的 4GB 内存。

如图 1.13 所示的那样，内存低端的 2GB(0x00 到 0x7FFFFFFF)供用户使用，内存高端的 2GB(0x80000000 到 0xFFFFFFFF)保留给内核。

1.3.1.2 缺页故障

创建一个进程后，Windows NT 执行部件把存储在磁盘上的可执行文件映象到进程的 4GB 虚拟地址空间中，并创建一个虚拟地址描述符。但刚开始时，并不从磁盘上将任何字节读取到 RAM 中，而是当试图存取文件的某个部分时才分配物理页面。

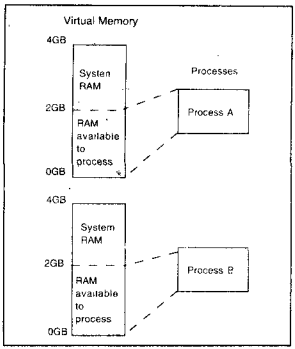

图 1.13 进程如何使用虚拟内存

可执行文件被映象到进程的地址空间后，如果应用程序试图存取这个文件中的某个位置，虚拟内存管理器就在 RAM 中搜索这个位置。如果被存取的地址所在的页面还没有读入内存，CPU 就发生一个缺页故障异常。缺页故障发生后，含有相应地址的页面便被装入内存。

当先前调入内存的页面不再需要时，操作系统以这个页面的所有者身份取消页面的控制权，释放其内存，供其它进程使用。

1.3.1.3 页面文件

当 VMM 执行磁盘缓冲操作时，用磁盘上的页面文件在硬盘和 RAM 之间调换文件。在磁盘和内存中，内存被划分为页面，即尺寸依赖于主机的内存单位，例如，在装配 80x86 处理器的计算机上，主页面尺寸是 4K 字节。

Windows NT 内核内存管理方式是对换物理内存页面和磁盘上的页面文件。当一页信息从磁盘移进物理内存，或从物理内存移到磁盘上时，只要进程受到这次移动操作的影响，