

## 第2章 重定向器

Windows使应用程序能通过操作系统内建的文件系统服务在网络上通信。有时候，我们称之为“网络操作系统”(NOS)能力。本章准备利用Windows 95、Windows 98、Windows NT、Windows 2000和Windows CE等均含有的Windows文件系统组件，向大家展示这些网络连接能力。本章的目的是让大家理解这些能力与邮槽和命名管道连网技术的关系。邮槽和命名管道连网技术分别是第3章和第4章的主题。

若应用程序希望访问本地系统中的文件，需要依赖操作系统来满足I/O(输入/输出)请求。我们通常把它称为“本地I/O”。例如，在一个应用程序打开或关闭文件时，需要由操作系统来决定如何访问包含了指定文件内容的一个设备。找到设备后，I/O请求会被转发给一个本地设备驱动程序。通过网络来访问一个设备也同样。然而，I/O请求必须通过网络转发给对应的远程设备。我们将其称为“I/O重定向”(I/O Redirection)。例如，Windows允许我们将一个本地磁盘标识符(如E:)映射或重定向到远程计算机上的一个目录共享入口。应用程序若指出自己需要使用E:时，操作系统便会将I/O请求自动重定向至一个设备，那个设备叫作“重定向器”(Redirector)。重定向器会建立到远程计算机的一个网络信道，以便访问指定的远程目录。随后，应用程序可自由使用一些常规的文件系统API函数，比如ReadFile(读文件)和WriteFile(写文件)等。虽然实际是通过网络访问的，但表面上却与访问本地文件无异。

本章着重讲解了如何通过重定向器将普通的I/O请求“重定向”到远程设备。本章内容对于以后的学习异常重要，重定向机制是邮槽和命名管道技术的基础。首先，我们打算解释如何通过网络，使用“多UNC提供者”(Multiple UNC Provider, MUP)资源定位符，通过“通用命名规范”(Universal Naming Convention, UNC)来引用远程文件。随后，我们讲解了MUP如何调用一个网络提供者，从而揭示出怎样通过一个重定向器，在“服务器消息块”(Server Message Block, SMB)协议的帮助下，在不同的计算机之间建立数据通信。最后，我们探讨了网络安全方面的一些问题。使用基本的文件I/O操作，通过网络来访问文件时，这些安全问题是必须考虑到的。

### 2.1 通用命名规范

“UNC路径”为网络文件及设备的访问建立了一套统一的规范。它最大的特点便是不必指定或引用一个已映射到远程文件系统的本地驱动器字母。这一点非常重要，因为应用程序可变得“与驱动器字母无关”。在复杂的网络环境中，应用程序不必对此有任何顾忌。同引用本地驱动器字母的方法相比，UNC名字要优越得多，因为在访问共享资源时，不必担心用光有限的驱动器字母的问题。另外，驱动器字母的分配也和具体的用户有着密切的联系——如果进程在你的用户环境中不能运行，便无法利用由你规定的驱动器字母映射关系。

UNC名字完全解决了这些问题，它的格式如下：

\\[服务器][共享名][路径]

第一部分是\\[服务器]，必须以两个反斜杠开头，紧跟着一个服务器名字。服务器的名字

代表着网络中的一台远程服务器，我们想访问的远程文件便位于其中。在 UNC 名字中，第二部分是[共享名]，它对应着远程服务器上的一个“共享入口”或者“共享位置”。所谓“共享位置”，实际就是文件系统中的一个目录（包括根目录），表示可共享的资源便放在这个位置下面，是其他机器获取共享资源的“入口”。而第三部分[路径]对应的是共享位置下的某个具体目录（或子目录）。举个例子来说，假定现在有一台名为 Myserver 的服务器，在其本地硬盘驱动器 D:\ 上设置了一个共享目录，名为 D:\Myfiles\CoolMusic，并将这一长串名字简化为“Myshare”这个易记的“共享名”。现在，假定该共享目录下含有一个名为 Sample.mp3 文件。那么，假如网络中其他任何一台机器想引用（访问）这个 MP3 音乐文件，只需像下面这样指定它的 UNC 名字即可：

\\Myserver\Myshare\Sample.mp3

可以看出，与其将一个本地驱动器映射到共享目录 Myshare，还不如通过网络用 UNC 名字来直接引用一个文件——因为所有机器使用的都是同样的 UNC 名字！

若通过 UNC 名字在网络中引用文件，应用程序便不必关心通过网络建立连接的细节，这显然是一种非常出色的设计。使用 UNC 名字，系统便可非常轻松地定位网络服务器共享目录以及文件路径。网络通信的所有细节都是由网络提供者的“重定向器”来负责控制的，本章稍后即会对此进行详细论述。完成了第 3 和第 4 章的学习之后，大家便会知道邮槽和命名管道技术非常依赖 UNC 名字。

在图 2-1 中，我们展示了 Windows 环境下在网络操作系统上建立 UNC 连接所需的一些常规组件。此图也显示了数据流在客户机与服务器的 NOS 组件之间逐渐推进的情况。以前面的 UNC 路径 \\Myserver\Myshare\Sample.mp3 为基础，本章将对每一个组件进行细致讲解，并阐述通过一个网络来打开这个文件的过程。

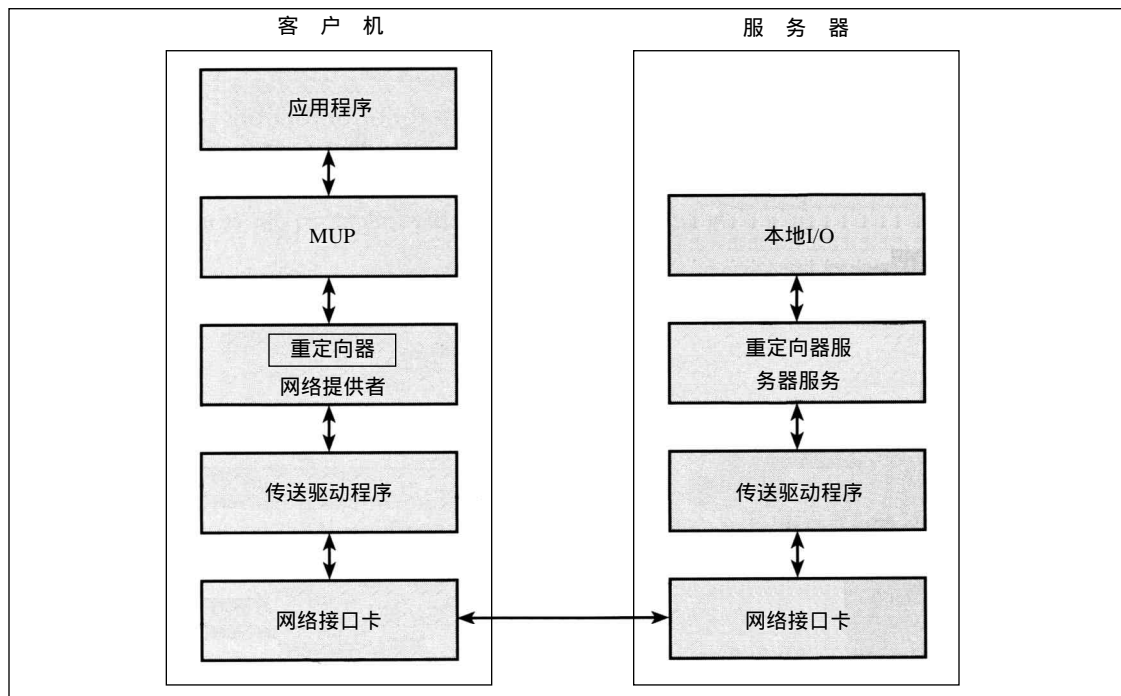


图 2-1 重定向器的各个组件

## 2.2 多UNC提供者

MUP如图2-1是一种资源定位器，负责选择具体的网络提供者，来满足 UNC连接请求。所谓“网络提供者”(Network Provider)，其实就是一种服务，可通过网络硬件访问位于一台远程计算机上的资源(如文件和打印机)。MUP需要通过一个网络提供者，在以 UNC名字为基础的所有文件及打印机 I/O请求上提供服务，建立通信关系。

Windows NT、Windows 2000、Windows 95以及Windows 98均支持多个网络提供者的安装。举个例子来说，Windows平台支持一个名为“Microsoft网络用户”(Client for Microsoft Network)的网络提供者。当然，亦可安装除微软以外的其他网络提供者，比如 Novell公司的 Novell Client v3.01 for Windows 95/98等等。换句话说，一次可能有多个网络提供者为一个 UNC请求提供服务。但必须注意的是，Windows CE目前仅支持一个网络提供者：“Microsoft网络用户！”

MUP的基本任务便是决定具体由哪个网络提供者来满足一个 UNC请求。为作出这个决定，MUP需将请求中提到的 UNC名字发给已经安装好的每一个提供者(以并行方式)。若某个网络提供者表明自己能够提供 UNC名字牵涉到的那一种服务，MUP便会将请求中剩余的部分发给它。如果有多个提供者都表示能够服务一个 UNC请求，MUP便会根据优先级，挑选出最恰当的一个提供者。网络提供者的优先级是由各个提供者在系统内安装的顺序决定的。在 Windows NT、Windows 2000、Windows 95和Windows 98中，要想对这种优先级进行改动，可考虑对注册表(Registry)中一个名为ProviderOrder的键值进行修改。它的位置在：

```
\HKEY_LOCAL_MACHINE
\SYSTEM
\CurrentControlSet
\Control
\NetworkProvider
\Order
```

根据优先级，已安装的各个网络提供者会按照先后顺序排列。由于 Windows CE只有一个网络提供者，所以根本就不会用 MUP来解析 UNC名字。相反，UNC请求会直接进入那个唯一的提供者。

## 2.3 网络提供者

如前所述，网络提供者实际只是一种服务，通过网络硬件来访问位于远程计算机上的共享资源，比如文件和打印机等等。事实上，这正是网络操作系统的一种核心功能。网络提供者具备的最主要的功能之一便是将本地磁盘标识符(如 E:)重定向至远程机器上的一个磁盘目录。作为提供者，它必须能为 UNC连接请求提供服务。在 Windows环境中，网络提供者需要向操作系统展示出一个重定向器，从而做到这一点。

Windows最有特色的一个网络提供者称为“Microsoft网络用户”，以前叫作“Microsoft网络提供者”(MSNP)。打开前述的注册表目录，排在第一名的往往便是“MSNP32”。通过 MSNP，可在 Windows NT 4、Windows 2000、Windows 95、Windows 98和Windows CE间自由地通信。但要注意的是，Windows CE不支持多个网络提供者，只提供了对 MSNP的内建客

户端支持。

## 2.4 重定向器简介

“重定向器”由网络提供者展示给用于接收和处理远程 I/O 服务请求的操作系统。要做到这一点，它需要格式化服务请求消息，再将其发给远程计算机的重定向器服务器服务。远程机器的重定向器服务器服务收到这个请求之后，会以发出本地 I/O 请求的方式，来满足这一请求。由于重定向器需要为较高级的服务（如 MUP）提供 I/O 服务，所以会在应用程序面前，将网络层的工作细节隐藏起来。这样一来，应用程序便毋需为重定向器提供协议特有的一些参数。因此，我们认为网络提供者是“与协议无关”的。换言之，在几乎任何网络配置中，应用程序都能正常运作。

MSNP 提供了一个特殊的重定向器，可直接与网络传送层和 NetBIOS 打交道，以便在客户机与服务器之间建立通信。本书第 1 章讨论的 NetBIOS API 函数提供了一系列编程接口，正好可以做相同的事情。由 MSNP 提供的这个重定向器有一个通俗的名字，叫作“LAN 管理器重定向器”（LAN Manager Redirector）。之所以叫这个名字，是由于当初设计它的时候，针对的便是老式的 Microsoft LAN Manager 软件，以便为 MS-DOS 应用程序提供网络操作系统能力（欲了解 NetBIOS 编程接口的详情，请参阅第 1 章）。NetBIOS 接口可通过大量网络协议进行通信。这便使得 MSNP 重定向器具有了“与协议无关”的特性，即应用程序不必关注某种网络协议的具体工作细节！若在自己的程序中使用了 MSNP 重定向器，便能通过 TCP/IP、NetBEUI 甚至 IPX/SPX 进行通信。显然，这是一种非常有价值的特性，因为无论网络在物理上是如何构成的，程序都可以正常地实现通信。然而，我们仍应留意一个重要的问题。两个应用程序要想通过网络相互通信，那么对两个工作站来说，必须有一种协议是“通用”的。换言之，两者必须至少安装同一种通信协议。举个例子来说，假定工作站 A 只安装了 TCP/IP，而工作站 B 只安装了 IPX。此时，尽管 MSNP 重定向器是“与协议无关”的，但仍会对此一筹莫展，无法通过一个网络在这两个工作站之间建立正常的通信。

MSNP 重定向器与其他工作站通信时，需要向对方的“重定向器服务器”服务发送消息。这些消息采用一种固定的结构形式，称为 SMB。至于重定向器具体如何收发消息，要由“服务器消息块文件共享”（Server Message Block File Sharing）协议来决定，或简称 SMB 协议。

## 2.5 服务器消息块

SMB 协议最早是由微软及 Intel 于 80 年代末期联合开发成功的。当时的设计宗旨是让远程的文件系统能由 MS-DOS 程序进行“透明”的访问。发展至今，这种协议的作用变得非常单一，就是以 SMB 数据结构的形式，让 Windows MSNP 重定向器与远程工作站的 MSNP 服务器服务进行通信。在 SMB 数据结构中，总共包含了三个基本组件：命令代码、命令特有的以及用户数据。

SMB 协议采用的是一个非常简单的“客户机请求 / 服务器响应”传输模型。MSNP 重定向器创建一个 SMB 结构时，需要在“命令代码”字段中指定一个特定的请求。若命令要求的是发送数据，比如一条 SMB 写指令，数据便会随结构一道传送出去。随后，SMB 结构会通过一种像 TCP/IP 这样的传送协议传给一个远程工作站的服务器服务。远程工作站的服务器服务会对收到的客户机请求进行处理，然后将一个 SMB 响应数据结构传回客户机。

现在，大家知道了通过 MSNP 重定向器建立通信时，需要用到哪些基本组件。接下来，且让我们看看假定通过一个网络打开 `\\Myserver\Myshare\Sample.mp3`，那么各组件的通信情况是怎样的。如下所示：

- 1) 使用 `CreateFile` 这个 API 函数，应用程序向本地操作系统提交一个请求，要求打开 `\\Myserver\Myshare\Sample.mp3`。
- 2) 根据从 UNC 路径描述中获得的信息，本地（本机）操作系统的文件系统判断出该 I/O（输入 / 输出）请求的目的地是一台远程机器，名为 `\\Myserver`，所以将此请求传递给 MUP。
- 3) MUP 调查出该 I/O 请求发给的是一个 MSNP 提供者，因为网上的 `\\Myserver` 机器正在使用 NetBIOS 名字解析机制。
- 4) I/O 请求随即传给 MSNP 提供者的重定向器。
- 5) 重定向器将此请求格式化成为一条 SMB 消息，要求打开包含在远程 `\Myshare` 目录下的 `Sample.mp3` 文件。
- 6) 格式化好的 SMB 消息终于通过一种网络传送协议，正式送入网络。
- 7) 名为 `\\Myserver` 的服务器从网上接收到这个 SMB 请求，并将请求传给服务器的 MSNP 重定向器服务器服务。
- 8) 服务器的重定向器服务提交一个本地 I/O 请求，希望打开位于 `\Myshare` 这个共享位置处的 `Sample.mp3` 文件。
- 9) 服务器的重定向器服务格式化好一条 SMB 响应消息，指出本地打开文件的 I/O 请求是成功，还是失败。
- 10) 通过一种网络传送协议，服务器的这条 SMB 响应消息返回客户机。
- 11) MSNP 重定向器收到服务器的这条 SMP 响应消息，并向本机操作系统传递一个返回代码。
- 12) 本机操作系统再将该代码返回给当初应用程序的 `CreateFile` API 请求。

从中可以看出，MSNP 重定向器必须经历大量步骤，才能让一个应用程序访问到远程资源。当然，MSNP 重定向器也需要对网络资源提供访问控制服务。这其实正是“网络安全”机制的一部分。

## 2.6 安全问题

这里对安全问题的讨论只限于通过网络对资源的访问。但在深入探讨如何保证通过网络安全访问一个资源之前，首先要对本地机器（本机）的安全机制有一定程度的了解。针对像文件和目录这样的系统资源，Windows NT 和 Windows 2000 都同时提供了本地和远程访问控制的能力。在此，需要将这些资源看作需要“保密”的对象。若应用程序试图访问一个保密对象，操作系统便会检查该程序，看它是否有那个对象的“访问权限”。访问权限共分几种，最基本的包括读、写以及执行。Windows NT 和 Windows 2000 是通过“安全描述符”以及“访问令牌”来实现访问控制的。

### 2.6.1 安全描述符

对需要保密的所有对象来说，都应包含一个特殊的“安全描述符”（Security Descriptor），规定自己特有的访问控制信息。在一个安全描述符内，包含了一个 `SECURITY_DESCRIPTOR`



结构，以及对应的安全信息，包括：

所有人安全标识符 ( Security Identifier , SID )：代表对象所有人。

组SID：指定对象的主组所有人。

授权访问控制列表 ( Discretionary Access Control List , DACL )：指定能由哪些人进行什么类型的访问。访问类型包括读、写及执行权限等。

系统访问控制列表 ( System Access Control List , SACL )：指定需要为哪些类型的访问企图生成审核记录，以便将来稽查。

就程序本身来说，无权直接对一个安全描述符结构的内容进行修改。然而，我们可利用一些 Win32 安全 API 函数，来进行这种形式的修改。通过这些 API，可进行安全信息的设置与获取。在本章结束的时候，会向大家展示具体如何操作。

#### 1. 访问控制列表和访问控制实体

安全描述符的 DACL 和 SACL 字段指定的是访问控制列表 ( access control list, ACL )。这些字段可能包含了零值，或包含着更多的访问控制条目 ( access control entities, ACE )。每个 ACE 都负责控制或监视着指定用户或用户组对一个对象的访问。在一个 ACE 中，包含着下述类型的访问控制信息：

一个 SID，标识着该 ACE 应用于哪个用户或用户组。

一个掩码，指出象读、写和执行这样的访问权限。

一个标志，指出 ACE 所属的类型——允许访问、拒绝访问或者系统审核等。

注意系统审核 ACE 仅在 SACL 中使用，而允许访问和拒绝访问这两种 ACE 类型在 DACL 中使用。在图 2-2 中，我们展示了一个文件对象的 DACL 设置情况。

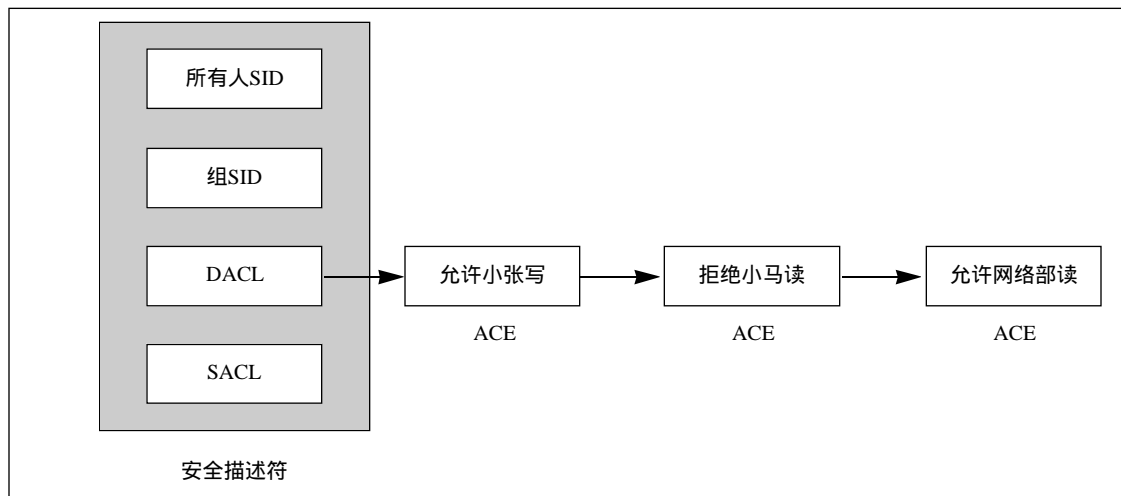


图2-2 设置了一个文件对象

对一个保密对象来说，假如没有 DACL（它的 DACL 已被 API 函数 SetSecurityDescriptor Dacl 设为一个空值），那么系统就允许所有人拥有对它的“完全”访问权限。若对象有一个 DACL，便会参照 DACL 中的 ACE 设定，只允许进行那些明确允许的访问。假如 DACL 中没有设置 ACE，那么任何人都不能以任何形式访问该对象。类似地，假定 DACL 设置了一些“允许访问”ACE，那么凡是 ACE 中未包括进来的所有用户及用户组，系统都会拒绝他们的访问。

大多数情况下，我们只需指定“允许访问”ACE就足够了。但下面这种情况是个例外：如果为某个用户组设置了一个“只允许访问”ACE，那么可能还要设置一些“拒绝访问”ACE，将那个组的某些特定成员排除在外。要想达到目的，必须将一个用户的“拒绝访问”ACE放在整个组的“允许访问”ACE之前。注意ACE的顺序是至关重要的，因为除非访问被允许或拒绝，否则系统会一直按序读取ACE。用户的“拒绝访问”ACE必须放在最前面；否则的话，在系统读到用户组的“允许访问”ACE时，便会为那些受到限制的用户开放不该开放的权限。

图2-2展示了如何设置DACL，以便向一个名为“网络部”的用户组开放读权限。假定在这个组内，包含了小安、小张和小王三名用户。我们的目的是为组内除小安之外的所有人都授予“读”权限。为做到这一点，必须在为整个“网络部”设置“允许访问”权限之前，先为小安设置“拒绝访问”权限。在图2-中，我们也为小张设置了一个允许访问ACE，允许他拥有“写”的权力。记住应用程序并不能直接操作ACL，它们必须通过一系列涉及安全保密的API函数，来做这些事情。

## 2. 安全标识符

我们知道，在保密对象的安全描述符及ACE设置中，包含了一个安全标识符（SID）。所谓SID，其实是一个独一无二的值，用于标定一个用户帐号、一个用户组帐号或者一次登录会话。对于像Windows NT服务器域这样的安全总监来说，它们需要在一个安全帐号数据库中对SID信息加以维护。用户登录进入后，系统便会从数据库中获得用户的SID，并将其置于一个用户的“访问令牌”中。以后凡是在牵涉到Windows NT安全的地方，系统都可根据令牌中包含的这个SID，正确判断出用户的身份。

### 2.6.2 访问令牌

用户登录进入一个Windows NT系统后，系统会对用户的帐号名和密码进行验证，两者统称为“登录凭据”。若用户登录成功（即验证通过），系统便会创建一个相应的访问令牌，并将该用户的SID分配给它。对面向这名用户执行的任何进程来说，都会拥有该访问令牌的一个拷贝。若一个进程试图访问一个受到保护（即保密）的对象，访问令牌中的这个SID便会与DACL中分配给SID的访问权限进行对比。

## 2.7 网络安全

到目前为止，我们已简要解释了如何在本地机器上实施安全机制。接下来，打算来看看通过网络访问一个受安全保护的對象时，如何确保安全。如早先所述，MSNP重定向器负责着不同计算机之间的资源访问。此外，MSNP重定向器还要通过创建用户会话凭据的形式，在客户机与服务器之间建立一条安全通信链路。

### 会话凭据

共有两种类型的用户凭据：主登录凭据和会话凭据。若用户坐在一台工作站面前，登录进入机器，那么他/她的用户名和密码便成为主要的凭据集，并保存在一个访问令牌中。对任何用户来说，在任何时刻，只能存在一套主登录凭据。若用户尝试建立与一个远程资源的连接（无论通过映射驱动器的方式，还是用UNC名字加以连接），便会对用户的主凭据进行验

证,判断他/她拥有对远程资源的何种访问权限。注意对 Windows NT和Windows 2000来说,用户可选择提供一套不同的凭据,以便在针对远程资源进行验证的时候使用。若用户提供的访问凭据是有效的,MSNP重定向器便会在用户的计算机同远程资源之间建立一个会话。重定向器会将会话与会话凭据关联到一起,后者包含了用户计算机用于对远程资源连接进行验证的那些凭据副本。在用户计算机与远程服务器之间,一次只能建立一套会话凭据。若机器 B提供两个共享位置(共享点):\Hack和\Slash,而且假如机器 A的用户将\Hack映射成G,并将\Slash映射成H,那么两个会话都会共享同样的会话凭据,因为它们引用的都是同一个远程服务器。

MSNP重定向器服务器服务负责着远程服务器上的安全访问控制。若 MSNP重定向器服务器试图访问一个受到保护的對象(保密对象),便会用会话凭据来创建一个远程访问令牌。从此时开始,对安全机制的管理便和在本地机器上别无二致。在图 2-3中,我们展示了MSNP重定向器如何通过Windows NT域安全机制,来建立安全凭据。

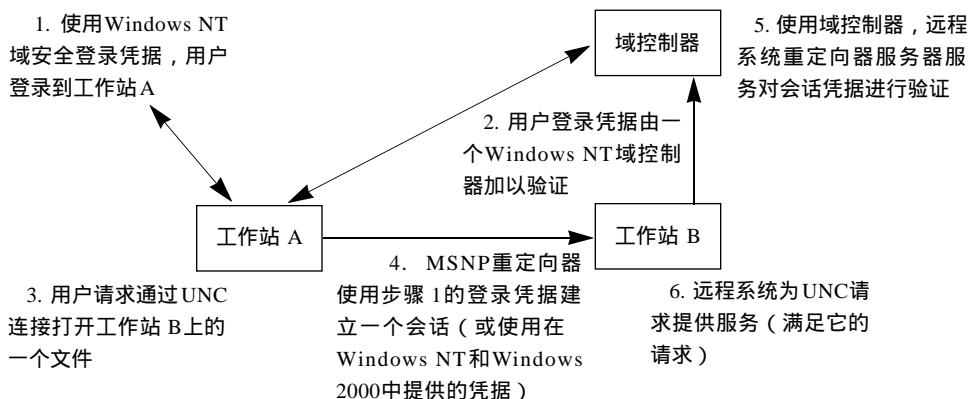


图2-3 安全凭据的演示

## 2.8 一个实例

使用MSNP重定向器,通过一个网络,Win32应用程序可分别使用CreateFile、ReadFile和WriteFile这三个API函数,来创建、访问以及修改文件。Windows NT和Windows 2000是目前支持Win32安全机制的唯一平台。在程序清单 2-1中,我们演示了如何编写一个简单的应用程序,通过UNC连接来创建一个文件。这些代码可在配套光盘上找到,位于\Examples\Chapter02目录下。

程序清单 2-1 简单的文件创建示例

```
#include <windows.h>
#include <stdio.h>

void main(void)
{
    HANDLE FileHandle;
    DWORD BytesWritten;

    // Open a handle to file \\Myserver\Myshare\Sample.txt
    if ((FileHandle = CreateFile("\\\\Myserver\\Myshare\\Sample.txt",
```



```
    GENERIC_WRITE | GENERIC_READ,  
    FILE_SHARE_READ | FILE_SHARE_WRITE, NULL,  
    CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL))  
    == INVALID_HANDLE_VALUE)  
{  
    printf("CreateFile failed with error %d\n", GetLastError());  
    return;  
}  
  
// Write 14 bytes to our new file  
if (WriteFile(FileHandle, "This is a test", 14,  
    &BytesWritten, NULL) == 0)  
{  
    printf("WriteFile failed with error %d\n", GetLastError());  
    return;  
}  
  
if (CloseHandle(FileHandle) == 0)  
{  
    printf("CloseHandle failed with error %d\n", GetLastError());  
    return;  
}  
}
```

## 2.9 小结

本章向大家简介了 Windows 重定向器。利用它，应用程序可通过网络来访问远程的 Windows 文件系统资源。我们首先解释了重定向器在网络上的基本通信方法。随后，讨论了在应用程序使用重定向器时，由 Windows NT 和 Windows 2000 提供的一系列安全特性。下面两章将分别讨论邮槽和命名管道这两种非常有用的技术。在其涉及到的所有网络通信中，邮槽和命名管道都要依赖于“重定向器”。