第四部分

开发者眼中的

Visual C++

返回总目录

目　　录

第 14 章　安全性设计

网络管理员在极力推行网络安全性时，将要面临很多问题，我们都知道：使用公用的或太短的保密字、用户不懂得安全操作以及对管理的完全忽视，都将直接威胁局域网的安全。幸运的是，由于有了类似 Windows NT 这样的操作系统，使得我们无论在本机还是在局域网内实施安全性时，都变得非常容易。学会如何使用这些安全特性并将它们置入你的应用程序中，对确保数据和应用程序的安全都是非常重要的。

在建立 Internet 站点或是 Intranet 站点时，安全性也是人们争论的主要焦点之一。公司可以通过建立一个 Internet 站点来获得公共的访问，也可以建立 Intranet 站点来尽量隐藏其身份，但现在很难知道哪一种会更好。不管你工作在什么样的环境中，市场上已经有很多公司开发的各类工具，以满足那些需要保护他们的数据不受损害的用户。然而，令人遗憾的是，当你建造自己的控件和应用程序时，所有这些华丽的工具实际上并不能从根本上保护你（当你用错误的方式使用它时，这些工具也不能很好地工作）。再次强调，学会 Windows NT 提供的安全特征，比如 ActiveX，将会帮助你提高远程访问技术的安全性。

在这一章中，你不应该去寻找那些定位在你特定工作环境中的具体答案。如果想在一章之中囊括当今所有的实用技术，或甚至囊括你曾遇上的每个安全问题，那都将是非常愚蠢可笑的。我们的重点是：从编程者的角度来看待安全问题，并讨论那些对编程人员实用的技术，但我们不针对具体的问题给出答案。

你最后所决定采用的方案应基于自己公司的需要以及你所使用的工具。

虽然 Internet 的安全性固然重要，但从整体上看，事实上，我们将更加集中精力关注 Windows NT 的安全性。我们也将关注那些使用远程访问所必须精通的技术，如 ActiveX 控件、分布式组件对象模型（DCOM）、组件对象模型加（COM+）等。这一章将要解答的一个问题是，编程人员可以采用什么手段来防止数据和应用程序代码受损？从编程者的角度来看，一个 ActiveX 控件不止是一个新的浏览器资源，它也可能是一个潜在的病毒源。在保护控件方面，除非是浏览器、编程者和 Internet 站点允许你更好地使用局域网资源，否则哪怕你做出了最仔细的努力，也将是枉费心机。这也意味着你将有可能因为一个笨拙的用户而暴露了自己的服务器，进而导致潜在的破坏。另外，因为 COM+还是一个海市蜃楼，所以你在使用这个新技术时应该特别小心。

微软为编写各类应用程序提供了各种 API 和开发工具，我们要做的第一件事就是花点时间学习一下这些 API，其中包括远程的和本地的，你至少应该看一点安全方面的 API，尤其是要看看类似 Windows NT 这样的产品所提供的本地安全性。既然你已经完全使用了操作系统，为何不充分利用它所提供的安全性呢？在一般的应用环境中，操作系统安全性带来的好处对任何人来说都是显而易见的（除非你正关注的是 Internet）。请牢记，你可以使用 ActiveX 控件来做一些事，如巧妙地处理注册登记或至少是用来询问用户和公司名称之类的信息，如果你可以使用 ActiveX 控件从注册中获取用户名和公司名，那么通过微软提供的操作系统服务，你也可以使用它来辅助实现其安全性。不难看出，你在 Internet 站点上所附加的那些安全性，将会分担很大一部分的安全压力。

我们在这一章要关注的另一个主题是：编程者在安全性方面所必须考虑的各类问题，也就是了解有谁进入了 Internet 站点、是如何进入的？了解你的组织结构中的脆弱部分，以及随意从 Internet 下载控件将会产生什么样的危害，这些都是非常重要的。要实施安全性，你首先应该知道你所保护的对象是谁、你所防止的东西又是什么？否则你的努力从一开始就是做无用功。

我们抛开了那些对 Internet 安全性的世俗顾虑后，将开始关注如何去实现它的安全性。这里有个问题，当 Sun 公司开始创建 Java 小应用程序（Java applets）时，它在安全性方面采用了一种称为“Sandbox（沙箱）”的方法，即一个 Java 程序只能在它自己的 Sandbox 中使用，如果越过了 Sun 规定的这个严格区域，程序将不能访问操作系统，或者甚至不能访问硬件系统。为了使用 Java 小应用程序，你需要一些运行策略来确保访问是被严格控制的，这种方法的好处非常清楚：越少的访问意味着越少的安全顾虑。遗憾的是，它也严酷地束缚了 Java 处理工作的能力，这尤其表现在不同类型的数据要进行交换之时。

注 Java 在安全性方面采用了“Sandbox”的方法

Web 链接 即使 Java 也不是“防弹”（bulletproof）的，黑客已经发现了它执行代码中的漏洞，这使得 Java 也并非是一个完美的解决方案。例如，在 1996 年 3 月 22 日，在普林斯顿大学计算机科学系的 Drew Dean（ddean@ICS.Princeton.EDU）和 EDFelton（felten@ICS.Princeton.EDU），发现一个 Java 臭虫（bug），这个臭虫允许他们创建一个程序来删除用户本地盘的某个文件，他们通过 Netscape 的缓存机构下载一个文件到用户的机

器上，然后让 Java 来执行这个程序，并以此来嘲弄 Java，幸运的是，在 Netscape 的 2.01 版本中已经修复了这个漏洞。通过访问 http://www.cs.princeton.edu/sip/pub/secure96.html 站点，你可以找到有关 Java 安全问题的其它信息。

微软提出的另外一个方案是 ActiveX 控件，它被称作是“shrink-wrap”方法，使用 ActiveX，就像你可以使用软件的任何一个 shrink-wraped 片一样，你也可以完全访问操作系统。这意味着，你可以写出一些非常灵活的控件，他们可以充分利用系统提供的资源（这里也没有任何来自从访问数据方面的限制）。然而，从安全的角度来看，ActiveX 也能导致恶梦一样的恐怖，有什么东西能够保护你下载的控件，并防止它对你的系统进行破坏呢？微软还有另一个手段：厂商认证。当你在某个商店买下一个软件时，你便能知道它的制造公司，如果软件中有病毒甚至有险恶的臭虫，你也知道和谁联系，微软提出的 ActiveX 控件的认证方式也和你在商店买的软件一样，它的包装也确切地标识出了软件的生产厂家。

Web 链接 所有同时使用 VBScript 和 JavaScript 的开发厂家正持续不断地提高这两种脚本语言的安全性。不幸的是，要发现 VBScript 的未来发展方向还有点困难，通过访问 http://www.osf.org/~loverso/javascript/站点，你能够读阅读有关 JavaScript 的著名安全问题。

脚本语句是很容易被监视的，我们现在关注一下运行脚本语言的安全性。

不论是 VBScript 还是 JavaScript，他们的近期版本都不是非常安全的（虽然已做了许多工作来寻找 JavaScript 的漏洞）。下文介绍了三个最常见的问题。

欺骗用户上载某个文件。尽管 JavaScript 在上载一个文件时，必须征得用户的同意，但黑客却能够通过多种途径来绕过这种限定，黑客真正需要的只是一个含有趣味标题的按钮，对使用 Windows95/98 的用户来说，上载他们的用户保密字文件是非常容易的，这使得黑客闯进你的操作系统变得更加容易。

获得文件的目录。在上载你机器中的一个目录时，JavaScript 没有要求必须征得用户的同意，事实上，它也可以上载其它网络驱动器中的目录。黑客如果知道了你硬盘的组织结构，便能够更容易地闯入你的系统了。

跟踪访问的站点。黑客通过跟踪你访问的站点，便能够对你进行一些识别，而 JavaScript 使得黑客更容易这样做，它能够跟踪你访问的每个 URL，并将它们的地址发送到黑客的机器中。就像上载文件时出现的问题一样，用户必须给出许可，而黑客却几乎可以用任何一个东西来掩码这种许可。

本章也将关注一下安全标准，一些编程者把这些标准当成累赘，因为它们与编程没有直接的关系。我们并不在这些标准的全部内容上浪费时间，但是知道它们的存在以及如何找到它们，也是非常重要的。你们中许多人也都认为，没有什么理由要对一个已经安全实用的东西再作重新改造。如果有人已经发现了一种能防止访问中受到危害的方法，而且这种方法所有人都可以采用，那有何理由不采纳这个标准的方法呢？而这些也就是在安全标准这一节中所关注

的，它让你知道这些标准中什么东西有用，且让你学会如何去使用它们。

14.1 精通 Windows 的安全 API

Windows NT 提供的安全级别对一个操作系统来说，几乎有点倾向于偏执，它允许你用很多的方式来设置安全性，包括既是日常用户又是文件级别的用户。你也可以创建用户组，并通过组来分配安全性，而不是对单独的用户来分配安全级别。另外，你也可以通过使用各种警报和登记文件来监视系统的安全情况。Windows NT 擅长于实时在地监视系统的活动情况，任何事件都逃不出其审查的范围，事实上，即便是从一个进程到另一个进程的信息传送，都要经过系统某一级别的审查。

对应用程序来说，Windows NT 提供的安全级别却是一把双刃剑。一方面，应用程序不能做太大的破坏，大多数情况下，在造成安全破坏之前，Windows NT 便能够简单地中断错误的应用程序；另一方面，对一些在 Windows95/98 中运行很好的应用程序，施加这么严格的安全限制，将会对程序的运行造成破坏。Windows NT 提供的安全性，从本质上说，确实影响了你的机器本身所能提供的兼容性。

技巧 如果只是想设计一个一般的应用程序，而没有什么与安全性相关的服务，你也许潜意识想用 Windows95/98 来做。可另一方面，Windows NT 提供了那么多的安全服务，以至于你可以创建一个界面几乎不能让人接受的应用程序，然而，要付出的代价是，你的应用将不能在许多机

器上运行，因为 Windows NT 和 Windows 95 在安全性上几乎处于两种完全不同的极端——Windows NT 具备了几乎太多的安全性，而 Window 95 却几乎没有。

Windows NT 提供的大量安全特性使 Internet 站点和用户得到了极大的便利，它的本地安全性使 Windows NT 成为了 Web 站点中的优秀平台。另外，如果你在使用 Windows NT 提供的标准特性来编写代码时遇上了什么麻烦，你也可以改进这些特性。例如，在第 13 章中，你能编写一个 ISAPI 过滤器来监视后台的事件，如果将 ISAPI 过滤器和 Windows NT 的安全特性结合起来，便能够提供至少在跟踪重要安全事件中所需要的任何东西。也就是说，即便你不能阻止黑客对你的安全性进行破坏，你至少还能够跟踪黑客做了些什么，以将其破坏减至最小。另外，识别黑客闯入的地方，也将是提高安全性的一种方法。

安全特性并不局限于 ISAPI 过滤器或桌面应用，你也能够将 Windows NT 的安全性纳入 ActiveX 控件或扩展 ISAPI 中。例如，你可以编写一个 ActiveX 控件，并用它来检测登录你的系统的用户是否真正获得了访问许可。你甚至还可以编写一个扩展 ISAPI，它可以根据系统允许用户的不同访问级别，发送给用户不同类型的 HTML 网页。

从理论上说，你也可以将 Windows NT 的安全性加入 DLLs 中，通过 DCOM 的使用，它们能被客户机器所访问（甚至可以是 Windows 95/98 的客户）。因为 DLL 将在 Windows NT 服务器上运行，而不是像常规一样下载到客户端，所以客户实际上受到了服务器安全特性的保护。实践证明，在应用中加入此类安全特性，并不增加多少网络负载和服务器处理时间。相比之下，你当然宁愿通

过 DCOM 实施安全性。

既然 Windows NT 已经提供了这么多的安全性，那么为什么还要在你的应用中努力去创建附加的安全性呢？第一个原因是，目前为 Internet 安全性设计的 API 标准还存在着漏洞，例如，我们已经说过的 Internet 部件下载服务的漏洞（在第 8 章中，我们可以看到 Internet 下载过程的部分论述），因为当前市场上的各种浏览器提供的安全防线还存在着漏洞，所以一些神通广大的人可以通过各种手段绕过我们的安全防线（我们将集中讲述 Internet Explorer，当然 Netscape 和其它浏览器也肯定存在这样或那样的毛病）。这些安全漏洞并非存在于 API 本身，而是存在于过去为 Internet 设计的一些解决方案中。实质上，这些过去设计的解决方案到现在已经过时了。以下将告诉你 Internet 部件下载服务中的三个漏洞。（当然，肯定还有其它漏洞）：

HTML<A HREF>标志。人们有办法通过使用<A HREF>标志来下载和运行一个可执行文件，在 Internet Explorer 的 3.x 或 4.x 版本中，对代码的检测过程中存在这个问题，它当前的做法是，使用 URL 名字直接下载代码，然后调用 WinVerifyTrust 来检测代码的合法性（这与第 8 章中的描述一样），这种方法 100%安全吗？回答是否定的，因为你使用了标准进程以外的一些东西来检验文件的内容，在这里你将依赖于这个 HTML 分析器。

脚本。现在的脚本对所有安全检测来说，是完全自由的。没有什么方法能检测出是谁创建了这些脚本，或它将会对你的机器做些什么？更严重的是，没有什么方法能检测出这些脚本将会从你的机器中取回些什么信

息！（在这一章开始，我们看到了 JavaScript 能从你的机器中取回的两种信息：用户硬盘中的目录和用户访问的站点）。微软正在创造一些脚本的认证，一旦脚本认证变成了现实，浏览器便可在运行代码前先调用 WinVerifyTrust 来检测这些脚本。

完整应用程序的下载，或其它更复杂的下载情况。IE 目前在为检测特殊类型的下载，做了许多很好的工作，例如，在下载一个 OCX 时将插入一个 WinVerifyTrust 进程。如果下载的参数超出了 IE 的检测限制范围，将会发生什么情况呢？例如，某一用户可能想下载并安装一个 Doom 或其它的游戏，而在安装中可能发生一些不可预料的事情，如申请注册或重启系统等，而此时的 IE 却不能控制它。微软正打算在以后组件下载版本中加强其健壮性，使它能控制这类事件。

Web 链接 作为保护策略中的一部分，你也应该测试多种浏览器并检测它们的反应（尤其在你创建一个公共访问站点时）。编程人员在与多种产品一起工作时，应该在头脑中形成一种基本观念，那就是不断地跟踪你访问过的所有 Web 站点，一种名叫 NavEx 的产品允许你将 IE 的收藏站点复制成 Netscape 的书签，反之，它也能将 Netscape 的书签复制成 IE 的收藏站点，通过访问 http://mach5.ocs.drexel.edu/navex/，你可以下载到这个产品。

注释 在这一章，我们也将探讨 Internet 技术中的其它漏洞。例如，表 14.1

列出了一些为填补这些漏洞而设计的新标准。如果没有出现安全问题，SHTTP、S/WAN 和其它类似的技术将是不必要的。MIME 有一种新技术叫 S/MIME，它能确保其它人无法偷窥你的电子邮件（mail）。

为了精通 Windows NT 是如何增强 Internet 站点安全性的，我们首先要来看看操作系统本身提供的安全特性。Windows NT 提供的所有安全特性，如映射网络驱动器，都能被应用程序所访问，许多编程人员好象忘记了，使用 ActiveX 控件也能够访问这些特性。例如，你可以像在第 10 章中创建一个按钮一样来调用它，其目的是显示一个网络映射对话框，以供用户选择一个驱动器来进行网络映射。

当然，Windows NT 也有一些安全特性并没有涉及 Internet 的安全领域。如映射网络驱动器，在应用程序中你可以通过创建一个按钮来调用它，但它并没有 Internet 的安全保护。然而，如果你在一个 ActiveX 控件中增加上 Windows NT 的一些其它安全特性，则可以实现它的 Internet 安全性。例如，你可以在一个 ActiveX 控件中增加上 Windows NT 的口令保护，这样，用户每次想要访问你的站点时，将显示一个登录对话框，要求用户输入用户名和口令。

在一个 Internet 的应用程序中，通过 ActiveX 控件来显示登录对话框，实际上完成了两件事：第一，Windows NT 提供口令保护的安全性比 Windows 95/98 要强，黑客如果没有猜出正确的保密字，将要周旋很长时间才能通过这个对话框；第二，你可以让服务器登记下所有的访问记录——请谨记，Windows NT 提供给了你监视任何事件的能力。万一实在有人闯入了你的系统，你至少也可以知道这个黑客使用了哪个帐户，并根据这个帐户的安全等级，评估黑客对你

的系统可能造成的破坏程度。

Windows NT 提供的安全性 API 函数将帮助你创建一个更加安全的 Internet 环境。要使用这些安全特性，在客户站点端，你可以使用 ActiveX 控件，而在服务器端，你可以使用扩展的 ISAPI，并通过在后台运行的 ISAPI 过滤器来监视事件的安全性。

14.2 使用 Windows NT 的安全性

在上一节，我们已经阐述了推荐使用 Windows NT 做 Web 服务器的一些理由，其最重要的一点是，能够使用 Windows NT 的安全性 API。然而，懂得为什么要存在 Windows NT 的安全性 API 只是一个起步，现在，我们将要关注的是，在现实中你是如何实施这些安全特性的。在这一章中，我们实际上讲述了五种不同类型或等级的安全特性，下面将对每种类型进行定义：

注 使用 Windows NT 作为 Internet 服务器的一个主要原因是，你能使用它的安全性 API。

内置安全性 作为一个操作系统，Windows NT 有一定级别的安全性。在 Windows NT 中，每个对象都对应着一定的安全性，我们将关注这些安全性的实施方法。使用对象级别的安全性意味着：如果没有正确的授权，几乎没人能有机会访问操作系统的任何部分或任何数据，当然，也不是一点机会也没有——我们总是假设你的安全性不是最完美的。

私有通讯技术(PCT) PCT 是微软和 IETF 一起创建的一种特殊的 Internet

安全性技术。在近期版本中，PCT将允许客户和服务器建立私有的通讯，以保证通讯不会被人窃听到，这种安全性的实现依赖于数字签名和加密方法。

Windows NT通过HTTP进行身份认证 许多人都错误地认为，使用这些安全性来工作将是非常繁琐和复杂的，但有时候，看似烦杂的东西却可以通过一个简单的方案做到尽善尽美。我们从Internet的角度来看，在Windows NT中进行身份认证有两种形式，一种形式是简单地询问用户的用户名和口令，并和服务器内的访问列表进行对照，这种形式又可通过两种方法实现，一种是在对Windows NT进行标准登录时进行的询问和回答；另一种则基于当前会话的设置，它也要求客户提供必要的用户名和口令，或者说，让用户使用客户提供的信息登录到机器上。另一种形式相对简单些，它依赖于一种叫安全套接层（SSL）的技术，SSL又是依靠加密和数字认证来工作的。如果你过于担心安全性问题，你甚至可以将这两种形式结合起来使用——它们彼此间并没有什么相互的制约关系。

数字签名 它是指发送人对他发送的文档或可执行文件进行签名，通过检测签名你可以判断其真伪。数字签名用一系列的私有或公开密钥来实现这个安全等级。

加密API 有些类型的安全性依靠多个保护层来工作，也就是说，如果一个黑客闯过了你的第一个安全保护层，则另一个保护层将阻止他进一步的访问。这就是加密API（也叫CryptoAPI）所有要做的，它是一个附

加的保护层，附加在所有已经存在的保护层上，它能保护你免受黑客更深层次的攻击。

在本章的下几节中，我们将阅读到几种新的技术，你将发现它既嵌入在Windows NT 中，也嵌入在新型的浏览器中。作为 Internet 服务器中的一种，Windows NT 支持如微软的 Internet 信息服务或对等网信息服务（在 Windows NT4.0 和 Windows95/98 的 OSR2 版本中免费提供）。我们将有趣地发现：和新版本的 Netscape 转接器一样，微软的 IE3.x/4.x 中也提供了内置 SSL 的支持；你也将发现，PCT 至少也支持 IE。我们也将关注数字认证码（即数字签名）技术，现在的许多软件厂家已经采用了它，它允许编程人员对其作品进行数字化签名，当你下载这些经过数字化签名的程序时，程序代码在签名后的任何损坏都将显示出来。它带来的好处是，你不会轻易下载到一个病毒感染过的程序。

Web 链接 通过访问 http://www.microsoft.com/iis/guide/pws.asp?A=2&B=5，你可以为你的 Windows 95 下载某个版本的个人 Web 服务器（PWS，Person Web Server）（Windows 98 的安装包中自带了 PWS），这个版本的 PWS 也与 Windows 95 的 OSR2 版本以及微软的 Windows NT 可选包随同销售（Windows 95/98 也有其可选包）。虽然 PWS 不会去满足制造 Web 站点的需求，但它能够让开发者很好地检测 Internet 应用程序。要知道，因为 Windows 95/98 没有提供与 Windows NT 一样的安全性，所以，如果使用 Windows95/98 的 PWS 版本，你将检测不到应用程序的任何安全特性。（幸运的是，在与 Windows NT 4.0 Workstation 一

同销售的 PWS 中，将允许你完全检测这些安全特性）。

加密也是保护你的数据的另一个可行方法，让我们参看一下下述微软已经开发的加密函数 CryptoAPI，这部分的技术至少是在 Internet Explorer 内的一种主要形式。然而，不管你现在能够使用它做什么工作，CryptoAPI 确实在程序内工作着，另外，对 CryptoAPI 的开发环境进行扩展，只在美国和加拿大是有效的。

内置的安全特性

在安全性方面，Windows NT 几乎是当前市场上最超前的操作系统。如果你对微软的这种地位还有什么怀疑的话，那么请去看看 Windows NT 获得的各种证书吧。你可以通过使用它的安全性能来获取很多便利，例如，你可以在一个 OCX 中使用这些高级功能，或通过 Internet 获得一些特殊功能的访问。在做这些事时，你必须受到一些限制，以免破坏你的网络安全。使用一些来自 Internet 的特性是不切实际的，因为它们并不重要。

幸运的是，ActiveX 控件却不必对 Internet 进行限制，没有什么条令规定，你不能在你的应用程序中创建一个有特殊用途的控件。在数据库或其它应用程序中的 ActiveX 控件，能被网络管理员（或其他有相同权利的人）正常地使用，这使应用程序的工作变得更加简单和可靠。事实上，如果你需要的话，ActiveX 控件也可以设计为多种个性化用途，你甚至可以给它增加检测控件当前位置的能力，或增加一个指定的位置作为特性页面配置的一部分，一些人可以选择其

用于 Internet 的特性子集，而另外一些人则可以选择其用于局域网（LAN）的特性子集，还有一些特性集则是用于本地的。

也许你们中许多人会问：需要什么类型的管理员来管理安全性，且不需使用网络操作系统（NOS）附加的工具呢？Windows NT 确实提供了大量的管理工具，而且这些工具都非常容易使用，所以一些编程者认为，再附加一些工具是不值得的。这个问题不好回答，但是，当大部分人在考虑 Window NT 下的安全性时，几乎总是这么想：如果管理应用程序的人不是网络管理员，而是练习使用网络操作系统的某个人，这又该如何处理呢？我们说这个人可能是一个工作组管理员或其他独立的个体，他没有必要看到整个网络图，而只需要足够的信息来维护他所负责的应用程序，你将发现这种情况是经常发生的，尤其是一个拥有许多小工作组的大公司，更是属于此类情况。尽管网络管理员不懂得正确管理应用程序，但也不愿让工作组管理员自由地访问整个网络。

注 ActiveX 在局域网环境的一个最好的应用是，它提供一定的子网安全管理能力。

不管你是在为 Internet、还是为局域网（LAN）或是为广域网（WAN）创建控件，你都将发现，懂得网络底层的安全构造是非常重要的。Windows 95/98 没有提供像 Windows NT 一样多的安全特性，所以，在 Windows 95/98 中，你有时会发现自己没有增加上应用程序的安全性。然而，尽管 Windows 95/98 没有为应用程序提供某个安全特性，但有时它们和 Windows NT 使用同样的安装程序，所以应用程序安全模块将同时为这两种操作系统工作（在其它情形下，你可以干脆为 Windows NT 使用单独的安全模块，以增强其安全性能，请看下

面的具体注释）。

注释 与 Windows 95/98 相比，Windows NT 确实提供了更多的安全 API 函数，因为它的安全性更加强壮。现实中，因为 Windows 95/98 缺少对 Windows NT 安全函数的支持，所以在使用它时，你会经常遇到一些障碍。例如，在 Windows 95/98 下，你不能调用 GetUserObjectSecurity 函数；而在下一节中，我们将会看到的操作访问令牌的函数，也大多不能在 Windows 95/98 中使用。弄清一个函数是否被支持的最好方法就是直接检测它，如果你在调用它时收到一个“ERROR_CALL_NOT_IMPLEMENTED(value 120)”的信息，则你便知道它只能在 Windows NT 中使用了。

在 Windows NT 和 Windows 95/98 中，“对象”（Object）这一名词的使用更加宽松。我们说，在事物的背后确实潜伏着许多对象，但你可能发现，“对象”这个名词的定义已经不能完全符合 C++的习俗了。在下几节中，我们把对象大体看成是：在运行一个指定的安全任务时所需的代码和数据的总和。或者说，每个安全对象自身包含了一个为充当某个角色而设计的单元。（在 Windows NT 和 Windows 95/98 的许多地方，微软主要选用 C++中“对象”的定义，因为作为 MFC 的一部分，它提供了所需的机能。然而，在阅读本章或微软的其它文档时，你不应该把对象看成是一个严格的 C++对象，而应该多加点 COM 的意识）。

知道了所有东西都是对象后，我们将更容易理解安全性——这至少是一个起点，当然，对象本身也就是一个起点。对象是被用户所访问的，所以在 Windows

中，安全性是指将对象所受的保护与用户的访问权进行比较。如果用户拥有足够的访问权（访问权等于或超过对象所受的保护），则用户能够使用这个对象。在 Windows 的文档中将对象所受保护的级别称作安全描述符（security descriptor），这是一种结构，它能告诉安全系统：用户需要什么样的权力才能访问这个对象；而用户则有一个访问令牌（Access Token），它是另一种结构，它能告诉安全系统：在一个给定的位置，用户有什么样的访问权。用户将访问令牌交给 Windows NT 系统，并以此获得对对象的访问。（如果把访问对象看作是一辆公共汽车，把 Windows NT 看作是驾驶员，则“车票”是访问令牌的一个很好的比喻，乘客只有出示车票才能搭车）。图 14.1 描述了这两种结构。

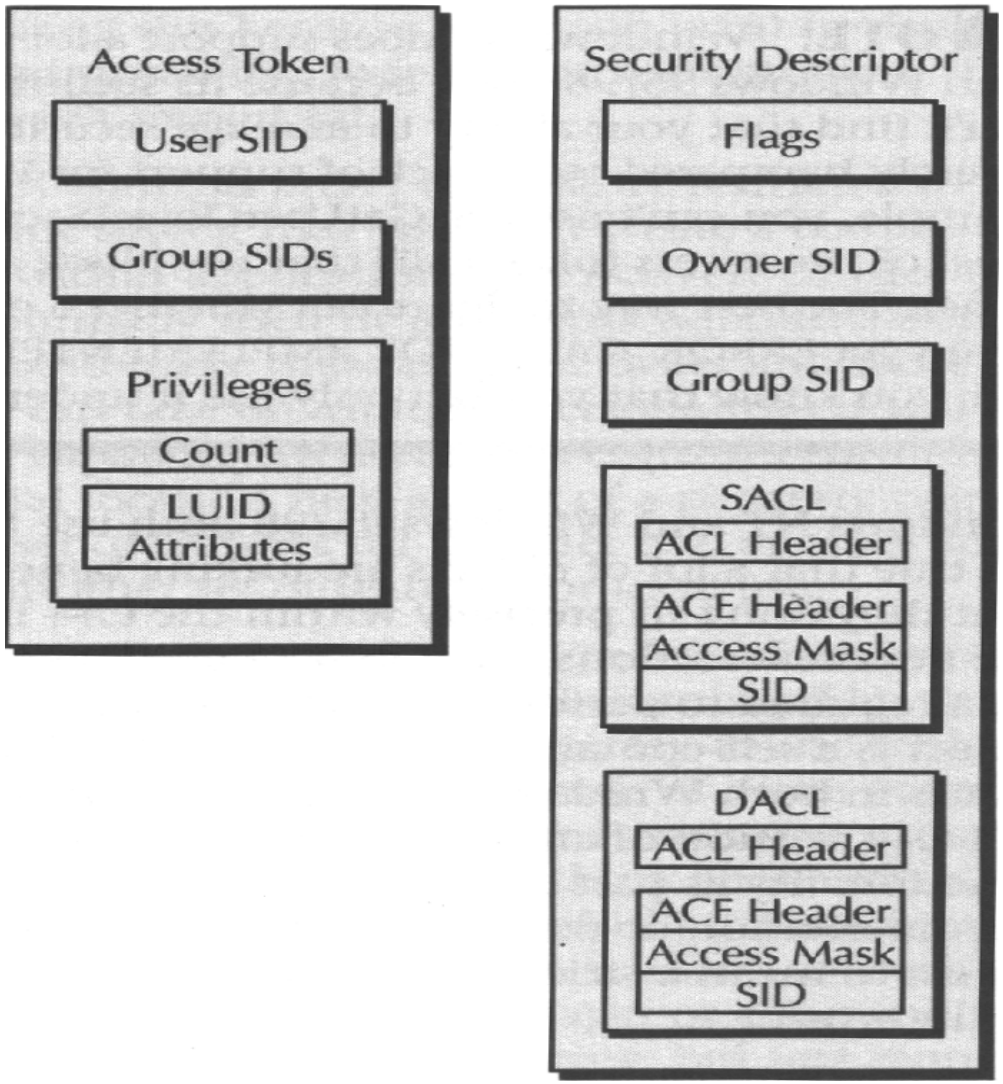


图 14.1　访问令牌定义了用户的权利，而安全描述符则定义一个进程的保护级别

以上，你只是稍微了解了一点 Windows 95/98 或 Windows NT 的安全性，简单地知道了它有安全对象和用户访问令牌，这与帮助你理解和使用 Windows

安全 API 函数还相差很远。在下面几节中，我们将细致地学习访问令牌是如何工作的；我们也将关注一下安全描述符。如果你只是对 Internet 感兴趣的话，在你使用 ActiveX 实施安全性时，可以完全不必懂得这些知识。然而，学会它，将会帮助你设计出更加符合心意的 ActiveX 控件。

理解访问令牌

在 Windows 中，你将发现有两种方法可以查看用户的权限，而且两者都是通过这样或那样的形式与对象关联的。用户的访问令牌有一个安全标识符（SID），它在整个网络中识别用户——它就像是一个帐户号。安全识别符表示了用户归属的组和拥有的优先权。每个组也有一个安全识别符，所以，用户的安全识别符也包含了他所属各个组的 SID，并将引用这些 SID。在 Windows NT 下，要想改变用户访问令牌的内容，你应该正规地使用用户管理器。

注 Windows NT 支持两种类型的安全标识符（SID）：用户 SID 和组 SID

那么访问令牌的优先权是指什么呢？首先是用户拥有特权数——不是用户所归属的组的数目，而是指访问令牌内指定的特权入口数，这部分还包括特权入口的阵列。每个特权入口还包括一个局部唯一的标识符（LUID）和一个属性掩码（Atrribute mask）。LUID 实质上是一个指向对象的指针，而属性掩码能表述用户对对象的权力。组的安全识别符也与此雷同，它们包括一个特权计数和特权入口的阵列。

技巧 现在，你可以去看看 Visual C++软件提供的 Windows API 帮助，找一找与安全标识符或访问令牌相关的 API 函数。例如，与 SID 相关的

函数包括 CopySID 和 AllocateAndInitializeSID；你会发现 OpenProcessToken 和 GetTokenInformation 函数也是非常重要的，它能在任何语言下保证应用程序安全正常工作。

高级技巧

观察访问权限传递

你还该知道的一件事是：有些类型的对象，如果它的权限没有被其它 SID 终止，则可能会继承到它的最低节点。例如，如果你让一个用户对硬盘的“\Temp”目录有读写的权限，则用户对该目录的子目录“\Temp\Stuff”也有相同的权限，除非你分配给这个子目录别的权限。对于容器对象也是如此，如果分配一个用户对容器对象的权限，如一个 Word 文档，则它允许用户查看这个容器下的所有内容，很多情况下甚至还有别的文件。由此看来，因为你可能会不经意地分配给用户过多的权限，所以使用安全审察来跟踪用户对服务器中各类对象的确切权限，是非常重要的。

使用访问令牌

因为访问令牌函数是你实施安全性前必须了解的第一个难点，下边，就让我们简要地谈论一下 Windows 提供的访问令牌函数。对用户的帐户做任何工作，哪怕是找出谁在访问指定的工作站，都必须了解访问令牌。正如前面所讲

的，访问令牌是安全等式用户方的中心部分。在对一个用户的帐户进行访问时，你几乎总要用到的一个函数是 OpenProcessToken，请注意这个函数的名字，它能够处理的对象可以是任何类型的进程、线程或用户等等，该函数的作用是获取附加在这个进程上的令牌句柄。例如，如果你查询一个用户的帐户，你必须有 TOKEN_QUERY 优先权，（你的访问令牌中必须包括系统委托的一些必要的权限，这就是为什么管理员能访问某个令牌，而其它用户却不能访问的原因）。而对用户的帐户要做任何改动时，你必须拥有 TOKEN_ADJUST_PRIVILEGES 优先权。还有许多其它的访问权限，我们在这里不再一一表述了。

一旦你拥有了一个访问令牌的句柄，你需要决定将对它做些什么。如果你想改变用户做某事的优先权，则你将需要这个优先权的局部唯一标识符（LUID），所有这些优先权在 WINNT.H 文件中都是以“SE_”开头的。例如，SE_SYSTEM_PROFILE_NAME 优先权允许你收集这个系统的配置信息。有些 SE 的值与用户无关（例如，SE_LOCK_MEMORY_NAME 特权允许进程锁定内存中的物理页面）。通过调用 LookupPrivilegeValue 函数，你可以获取对应局部系统上的优先权的 LUID。一般而言，你将使用 AdjustTokenPrivileges 函数来实现你所需要的修改。

注 在 WINNT.H 文件中，查看一下 Windows NT 定义的优先权列表，这些优先权是你能修改的。

查询用户帐户（或访问令牌的其它信息）是直接了当的。你可以调用 GetTokenInformation 函数来获取你需要的信息，这个函数需要一个令牌类参数，它把你所需信息的类型告诉 Windows 系统。例如，如果你想知道一个指定用户

的信息，你应该使用 TokenUser 类参数，你也将需要提供相应的结构，以让 Windows 保存你请求的信息，这一点与基于令牌类的请求不一样。

了解安全描述符

现在让我们来关注一下安全描述符。在图 14.1 中，每个安全描述符包括五个主要的部分。第一部分是一个标志列表，这些标志指明了描述符的版本号、格式和访问控制列表（ACL）的状态。

接下来的两部分包含了多个安全标识符（SID），所有者标识符表述了对象的拥有者，这不必是一个单独的用户；Windows 也允许你在这里使用组的 SID。有一点是受限制的，即组的 SID 必须出现在想要改变入口的人的访问令牌中。组标识符允许一组人拥有某个对象。这两种标识符中，只有所有者标识符在 Windows 下是重要的。组的标识符作为一个部分用于 Macintosh 和 POSIX 的安全环境。

最后两部分包含了各种访问控制列表（ACL）。安全访问控制列表（SACL）控制了 Windows 的审计功能，用户或组每次访问一个对象时，这个审计功能便启动了，Windows 在审计记录中产生一个登记项；全权访问控制列表（DACL）控制了谁能真正使用某个对象。你可以将一个组或用户分配给一个指定的对象。

注释 实际上，有两种型号的安全描述符：绝对安全描述符和自身安全关联描述符。绝对安全描述符在它的结构中包含了每个 ACL 的真正拷贝，这种类型的安全描述符用于需要指定句柄的对象；而自身关联安全描述符只包括了指向 SACL 和 DACL 的指针，这种描述符在改变组的

权限时，节省了内存并减少了所需时间，当某个特定组内，所有对象需要同一级别的安全性时，你应该使用它来处理，例如，通过这种方式，你能将单个进程内的所有线程赋予安全性。在你要保存安全描述符或将它传送给别的进程前，Windows 要求你把一个自身安全关联描述符转换成绝对安全描述符。通过 API 函数获取的描述符都是自身关联类型的，你必须在保存它之前将其转换。通过对 MakeAbsoluteSD 和 MakeSelfRelativeSD 两个 API 函数的调用，你可以在这两种类型的描述符之间转换。

一个 ACL 由两种类型的表项组成：第一个表项是个结构头，它列出了 ACL 包含的访问控制表项（ACES）个数，Windows 使用这个数字检测是否到达了 ACE 列表的末端（这里没有任何其它记录或手段能够检测每个 ACE 的准确大小）；第二个表项是个 ACE 数组。

警告 不要手工直接修改 ACL 或 SID 的内容，因为在以后的 Windows 版本中，微软可能改变它们的结构。Windows API 为改变这些结构的内容提供了丰富的功能。如果要对某种结构进行任何的改变，请通过调用 API 函数来实现，以免不必要的冲突。

那么什么是访问控制项呢（ACE）？ACE 定义了单个用户或用户组在一个对象上的权限。每个 ACE 由三部分组成：首先是一个结构头，它定义了这个 ACE 的类型、大小和一些标志；接着是一个访问标志，它定义了一个用户或组对这

个对象的权限；最后是用户或组的安全访问标识符（SID）的项。

一共有四类 ACE 头（其中有三类已用于当前的 Windows 版本中）。其中，access-allowed 类出现在 DACL 中，它认可用户的权限，你可以使用它增加用户对一个对象的权限。例如，虽然你想通过禁止用户修改系统时间，以使网络上所有机器的时间同步，但白天还是可以修改时间，而用户需要有这个权限，你可以通过使用一个 access-allowed 类的 ACE 来给予用户改变时间的权限；另一类是 access-denied 类，它废除用户对某一对象的权限，在一个指定的系统事件中，你能够使用它来拒绝对某一对象的访问，例如，当你对某一远程终端进行某些更新时，你可以取消其他人对它的访问权限；还有一类是 system-audit 类 ACE，它与 SACL 一起作用，它定义了指定的用户对哪些事件进行审查。当前还未使用的是系统警报 ACE，它允许 SACL 或 DACL 设置一个指定事件发生时的警报。

注 在当前的 Windows NT 版本中，使用的三类访问控制项（ACE）是 access-allowed、access-denied 和 system audit。

技巧 现在，你可以翻阅一下 Windows API 的帮助文件，看看 Windows 提供了什么类型的访问权限。为了获得这些信息，你应该更关注各种类结构，尤其是 ACL 和 ACE 的结构。看看 ACE 标志，它决定了在一个容器下的对象是如何作用的。例如，查一下 CONTAINER_INHERIT_ACE 常数，它允许子目录继承父目录的安全特性。

使用安全描述符

到现在，你只是了解了什么是安全描述符，以及它包含的各类结构是如何相互作用的，你还应该知道怎样开始一个实际的访问进程，并使用安全描述符来写一个程序。你必须首先了解的是：安全描述符不同于访问令牌，它没有普及开，所以你不能使用一个标准的函数集来访问它们。实际上，一共有五类安全描述符，每类安全描述符都使用不同的函数集来访问对象。（你必须拥有SE_SECURITY_NAME特权，才能使用这些函数）。

注 Windows NT支持五类安全描述符，每类都需要不同的函数集来访问描述符

文件、目录、管道和邮件槽。通过使用GetFileSecurity和SetFileSecurity函数，可以访问这种类型的对象：

注释 只有Windows NT下的NTFS文件系统提供了安全特性。Windows 95/98下的VFAT文件系统只提供了很少的安全特性。你不能从HPFS或FAT文件系统中分配或获取安全描述符。FAT文件系统没有提供任何扩展属性的空间，而这些空间是增加安全性所必须的；HPFS文件系统虽然提供了扩展属性，但它们没有任何安全特性。在上述的这些文件系统中，NTFS是最安全的。然而，不要错误地认为会有完全安全的文件系统。在Internet上，有一些应用程序能够读取NTFS分区的内容，即便用户没有正确地登录到Window NT中。

要访问进程、线程、访问令牌和同步对象，你必须使用

GetKernelObjectSecurity 和 SetKernelObjectSecurity 函数。所有的这些对象都是内核对象，为了更好地保护这些对象，它们也都有自己的安全描述符。

要访问 Windows 站点、桌面、窗口和菜单，你可以使用 GetUserObjectSecurity 和 SetUserObjectSecurity 函数。一个 Windows 站点包括键盘、鼠标和屏幕，即你用来访问系统的所有硬件。桌面包括窗口和菜单。这四个对象按此次序依次从前一个对象继承权限，也就是说，桌面将继承站点的权限。

要访问系统注册键，你必须使用 RegGetKeySecurity 和 RegSetKeySecurity 函数。注意，Windows 提供的所有注册类的函数都是以 Reg 开头的。

可执行服务对象，QueryServiceObjectSecurity 和 SetServiceObjectSecurity 函数用于可执行的服务对象。奇怪的是，在 Windows 的 API 帮助文件中，这两个函数都没有出现在其它安全函数的引用中。你在寻找这些函数之前必须已知道有这么两个函数。可执行服务是 Windows 提供的后台任务，如 UPS 的监视功能。通过双击控制面板中的服务程序，你将找到你的系统所支持的服务。

一旦你获得对某个对象的访问权限，你将发现，通过使用一般的 API 函数集，你便能够处理很多任务。例如，GetSecurityDescriptorDACL 能够从任何类的描述符中获取一份 DACL 的拷贝。换句话说，这些对象的描述符跟随一个同样的格式——即便大部分成分的描述符长度不同。导制这些长度不同的原因之一是，它们各自包含了不同数量的访问控制项（ACE），另外，安全标识符（SID）

的长度也不同。

要查询或修改一个安全描述符的下一步工作是分离这些成分。例如，通过使用 GETACE API 函数，你能够查看在一个 DACL 或 SACL 中单独的 ACE；通过使用与 SID 相关的函数，你也可以使用所有者标识符或组标识符（我们已经在本章的访问令牌部分阐述了这些函数）。可以满意地说，只要按指定的步骤做，你便能够使用一般的函数集来操作安全描述符。大致上说，对安全描述符的任何访问，一般都要经过下述三个步骤：

1. 获取描述符
2. 删除指定的成分
3. 修改指定成分的内容

要修改安全描述符，你应该按相反方向执行这些步骤。换而言之，你先使用类似 AddACE 函数来增加一个新的 ACE 至一个 ACL；然后，使用 SetSecurityDescriptorSACL 来改变一个描述符中的 SACL；最后，使用类似 SetFileSecurity 的函数（假设你想要修改一个文件对象）来保存安全描述符。

Windows 中的 ACEing 安全性

当你开始考虑 Windows 在 DACL 中计算 ACE 的方法时，你也许会发现一些潜在的存在问题的地方：虽然 Windows 的系统应用程序会自动关注这些问题，但你必须在自己应用程序中编制好了，才能运算出同样的结果。（SACL 也有同样的问题，但它只是影响审查，从系统安全的角度看，它不是那么要求苛刻的）。

Windows 是按照 ACES 在 ACL 中出现的次序来计算访问控制结果的。首

先，它可能不是一个很好算法，而且，在某些情况下它还会产生一些问题。例如，如果你想废除所有用户对某一区域的权限，但是在他或她的访问控制列表中包含了一个能够访问这一区域的组（即他或她是这一组的成员），这种情况的结果会如何呢？如果你将 access-allowed 访问项放在列表的第一位，则这个用户将能够访问这一区域——一旦 Windows 发现第一个满足用户所需权限的 ACE 时（或者一个禁止所需的权限的 ACE 时），它将停止继续搜寻。许可的权限是累加的，如果一个 ACE 允许对一个文件进行读的权限，而另一个允许写的权限，则当用户询问读和写的权限时，Windows 将综合这两个 ACE 并允许用户进行读写。下面的注中表示你应该将所有 access-denied 访问控制项放置列表前端，以防止安全上的潜在漏洞。

注 ACE 在 ACL 中排列的先后次序是非常重要的，因为 Windows 按次序对它们求值，并一旦发现第一个 ACE 允许或拒绝对某一对象的访问时，便停止继续搜索。

你也必须注意组的安全标识符（SID）的排列次序，因为用户从他（或她）所属的组获取的权限是相累加的。这意味着，如果一个用户属于两个组，其中一个组对某个文件有访问的权限而另一个没有，假如允许访问的组在列表中排列在前，那么用户可以访问这个文件，否则不能。

显而易见，为了将组排列成最好的次序，你将花费很多的时间。当用户拥有的组的权限和独立的权限数增加时，则可能出现无意识的安全漏洞。所以，仔细地创建一个组并严格限制用户单独的权限是非常重要的。

其它安全考虑

当你关注在 Windows 95/98 或 Windows NT 下的安全性时，还有两方面需要考虑：数据保护和服务器保护。前者对付客户的这种能力：当客户通过服务器（这里不是指文件服务器，而是某些类型的 DDE 或其它应用服务。）访问数据时，访问了那些没有获准访问的数据。我们用这种方式思考一下：如果客户对某一指定类型的数据没有访问权限，但是通过调用他有权访问的 DDE 服务器访问了这些数据，这将会有什么后果呢？服务器又应该如何保护它们的数据，而不是被迫成为造成安全破坏的同谋呢？

Windows 提供了一些 API 函数允许服务器模仿一个客户。大体上是：为了检测出客户是否对一个数据或进程有过多的权限，这些函数允许一个服务器假扮成客户所受的安全限制。例如，Windows 的一个 Word 用户可能需要访问 Excel 的一个数据文件，而这个用户可以通过 DDE 来获得这个文件的访问，在这种情况下，服务器需要检测一下这个用户是否有权访问这个文件，然后才传送所需的数据。当用户使用这种技术时，服务器甚至能找出客户的优先权。服务器唯一关心的是对它所管理的数据、资源和环境的保护。

这个 API 函数集支持三种不同类型的通信：DDE、命名管道和 RPCS。对每种通信类型，你必须使用不同的 API 函数。例如，要模仿一个 DDE 客户，你应该使用 DDEImpersonateClient 函数。Windows 目前在模仿这方面所提供的支持还有一些限制。例如，它现在还不支持 TCP/IP 连接，所以在这种情况下，你必须依靠别的手段来检测一个用户是否有合适的访问权限。

另一个安全考虑是如何保护服务器自身。在一个 Windows Word 用户调用

Excel 服务时，用什么方法来防止这个用户对 Excel 服务器本身进行破坏呢？请确信，对文件和其它有名字的结构，服务器自动会附加上一个安全描述符到这些对象上，并进行严格的保护（一个 DDE 服务器，如 Excel，在这种情况下不需要任何安全描述符，因为文件已在文件服务器的控制之下）。然而，大多数 DDE 或应用服务器的私有对象没有命名，它们需要另外的保护。Windows 也提供了一些 API 函数来帮助服务器保护自身。例如，CreatePrivateObjectSecurity 函数允许服务器附加安全描述符到它的任何私有对象上——如线程或其它进程。这些安全描述符能够防止除服务器外任何人访问其私有对象。

私有通信技术（PCT）

微软和 IETF 正携手开发一个名叫 PCT 的新的底层协议，和安全套接层（SSL）一样，PCT 是用来防止黑客窃听服务器和客户间的通信的，它通过加密、身份认证和数字签名技术来实现通信的安全性。由于可以使用 SSL，所以客户身份认证可以根据需要进行选择。

Web 链接 如果想了解有关 PCT 当前的进展情况，请访问 http://www.lne.com/ericm/pct.html，这个文档中包括了 PCT 的第二个版本的当前草案，另外，通过访问 http://www.w3.org/security/，你也可以参考一下 W3C 的 Web 站点上有关当前 Internet 安全技术的详细情况，其中包括 PCT 和 SSL。

PCT 假设你有一个可靠的传输协议，它能取代类似 TCP 的协议。有些人把 TCP/IP 看成是一种唯一的协议，但这是错误的，TCP 只是这些协议中的传输部分，而 IP 才是数据传输部分。IP 不支持任何形式的数据加密，所以，当你使用 TCP/IP 时，你的数据对任何想看它的人都是开放的。使用如 TCP/PCT 或 TCP/SSL 协议将使你的通信安全化。PCT 的第一个版本修正了在 SSL 中出现的一些问题，现陈述如下：

简化了的消息和记录结构。如果你没有使用客户身份验证，那么会话的重新连接在每个地址上需要一个唯一的消息；即便使用客户身份验证，重新连接在每个地址上也只需要两个消息。

扩展的加密算法。PCT 提供了比 SSL 更丰富的算法，这意味着它能支持更广阔的协议特性，而且这些协议特性的交涉是相互独立的。例如，公共特性包括加密类型、服务验证类型、散列函数类型和密钥字交换类型。

改进的消息验证密钥。在 PCT 中，消息验证密钥和加密密钥是相互独立的，这意味着，即便加密密钥是短的或不存在的，只要消息能使用一个很长的密钥，也能保证传输的安全，导制这一特性的主要原因是：因为美国政府把管理安全传输的密钥限定为最长为 40 位。

修补了安全漏洞。PCT 在会话中的客户身份验证是基于加密算法的。这防止了某些人通过捕获客户的验证密钥，并切断原客户的会话，而使用窃取的密钥来建立重新连接。客户只有知道了密文和密钥，才能获得服务器的访问。

附加的提前检验区域。客户和服务器在最初的握手对话过程中，其通信

是透明的，这个附加的区域使得客户和服务器可以检测在这种通信中可能出现的任何损害。

注释 虽然 SSL 在版本 3 中也提供了提前检验区域，但黑客也能够通过将协议版本号修改成 2 来绕过它，因为版本 2 中没有提供这一功能，而版本 3 完全兼容版本 2，所以客户和服务器都不会注意到这个变化。

微软当前正在开发 PCT 的第二个版本，这个版本与第一个版本完全兼容，但提供第一个版本所不具有的几个重要特性。以下是这些特征的简单概述。

新的数据报记录类型 单独的记录作为“数据报”独立地发送，从本质上看，这意味着协议不会保证数据发送的次序或到达目的地址的次序，这要求客户将接收到的数据进行排序，然后检验它们是否全部到达。这种方法的最大好处是提高了传输速度。

可重新识别的记录类型 记录的头包含了一些信息，它告诉接收者将接收什么类型的记录。

连续记录 PCT 版本 1 中允许数据跨越多个记录，即便是记录头没有指明连续的任何形式，版本 2 的记录头中增加了连续区，它也允许协议信息跨越多个记录。

数据记录的中间处理 数据记录是封装的，它允许发送者对数据做一些中间处理，比如压缩。

数据报记录的独立解密 由于发送的数据报记录可能会通过一个不可靠的传输通道，所以这部分特性对安全通信是非常重要的。每个记录分别加密，使得接收者可以立即对数据报解密，而不管接收数据报是否打乱

了次序。

新的密钥管理记录类型 这个记录类型允许发送者在会话中临时改变加密或消息验证密钥。重要的是，这允许了 PCT 来传送预先加密的数据。

新的关闭连接密钥管理消息 这是一个特定的消息，它告诉其它参与者关闭连接。因为这个消息是加密的，所以黑客很难发送一条伪装的消息来关闭这些连接。

增强型消息验证 消息验证现在也包括了记录头。

改进的握手协议 客户和服务器端的验证都包括了更加丰富的选项，其中有密钥交换、签名公共密钥和证书。

新的私有验证特性 这个特性允许客户和服务器使用预先一致共享的私有密钥进行验证，而不是使用公共密钥。

既然我们对 PCT 有了一些基本的了解，那就让我们看看 PCT 是如何工作的。PCT 使用可变长度的记录作为通信的方法，每个记录包括一个记录头，它定义了记录内的信息类型。一共有两类信息：应用信息和协议信息。应用信息总是包含数据，它们可以用标准的 PCT 格式或数据报格式；协议信息包含了密钥管理、错误或握手信息。PCT 使用两种附加的层，记录总是使用一个连接传送。客户和服务器间一般只有一个连接，但没有任何理由不能有更多的连接，每个连接只是会话的一部分。再强调一次，虽然你一般只能看到客户和服务器间的一个会话，但你确实可以用多个会话。（多会话可能需要客户和服务器间建立多个物理连接）。

PCT 协议的连接由一个握手信号开始，这就是握手管理消息类型开始工作

的地方。客户和服务器交换多个信息，首先是连接会话密钥的握手，或者说是客户和服务器决定一个用于相互对话的保密字。此时，客户和服务器也彼此进行验证。如果彼此认为对方不可信时，便不会进行这种握手；一旦客户和服务器都认为对方可信时，它们便约定好了一个用于信息加密的主密钥。

HTTP 上的 Windows NT 认证

Windows NT 的认证一般不是编程者要考虑的问题，更重要的是网络设置时要考虑的问题。但作为程序员，你确实应该了解一下认证问题，特别是如果你计划使用 Internet 或 Intranet 工作时更是如此。Windows NT 现在支持两种基本类型的认证。（虽然没有任何东西阻止一些人使用其它方法。）

技巧 如果你想使用 Windows NT 的质问/回答（challenge/response）特性，你必须使用 Internet Explorer 2.0 或更高版本的浏览器，其它浏览器当前还不支持这种实施安全特性的方法。

第一种认证的方法是 Windows NT 的质问/回答。这种认证方法依赖于服务器和客户密钥的通信，而没有用户输入的任何形式。在客户初始登录到系统时，服务器要求客户输入用户名和保密字，客户提供一个特殊加密的用户名和保密字，他还必须提供一个域名，因为客户必须在服务器的域内或在一个服务器所信任的域内。由于 Windows NT 的质问/回答在客户和服务器间自动使用加密通信，所以它比服务器提供的基本认证更加安全。

注 Windows NT 提供的两种形式的认证是基本认证和质问/回答，使用质问

/回答认证是最安全的。

Web 链接 有时，你顺手便可阅读到有关 Windows NT 针对 Internet 安全性方面的信息。其中一个信息源是 Rick 的 Windows NT 信息中心即 http://rick.wzl.rwth-aachen.de/cgi-bin/isindex3.cmd，这个 Web 站点涉及的一些内容，是你在微软的用户手册中所看不到的。例如，你将学会怎样使用 Windows NT 的各类第三方产品。在微软的方案中没有预先考虑到时，有些知识很值得学习。

那么客户是如何知道需要发送用户名和保密字到服务器的呢？服务器要求这些信息作为头的一部分，实际上，服务器发送了一个错误信息（401 Access Denied 即访问拒绝），它告诉用户需要一个安全访问。浏览器中实际上没有包括从服务器来的所有信息，懂得这一点是很重要的，因为你在浏览器中看到的只是服务器想要你看到的信息，它已经过浏览器筛去了头信息。例如，服务器一般必须告诉浏览器它正接收什么类型的信息，这样如果需要的话，浏览器便能够激活一个帮助。知道有这些头信息，对你了解在客户和服务器间的动态数据传输是非常重要的。

警告 如果你在你的站点上选择基本认证方式来实施安全性，你必须使用 SSL 来确保用户名和保密字传输的安全性。否则这些信息很容易被人破解，并利用它获得你的站点的访问。

第二种方法依赖于诸如 SSL 的数字签名技术。实质上，Windows NT 将需要一个从客户机来的数字认证，客户也需要 Windows NT 的数字认证。这些数

字认证可以从第三方厂家（如 VeriSign）获得，它能够保证双方的一致性。我们将会在本章的另一节中看到数字签名技术的确切过程。你现在应该知道的是，数字签名是一种安全手段，它能够标识试图访问你机器的另一方。

当使用 SSL 时，实质上是 Windows NT 服务器需要客户的标识。Windows NT 所获取的是由第三方厂家（如 VeriSign）发布的认证和随同的一个公共的密钥。SSL 在进行认证时，一共有以下六个步骤：

1. 客户发送给 Windows NT 一个没有加密的随机消息并伴随一个它的 VeriSign 发布的认证（它包含了客户的公共密钥），VeriSign 发布的认证是使用 VeriSign 的公共密钥加密的。因为每人都有 VeriSign 的公共密钥，所以 Windows NT 能够将整个认证解开并准确地检查它；同样，没有人能够伪造一个他们自己的公共密钥——他们必须从 VeriSign 那里获得。

2. 一旦 Windows NT 确认它从客户那里接收到了一个有效的证书和公共的密钥，它将告诉客户发送一个它最开始发送的消息的加密版本。

3. 客户计算机计算原始随机消息的摘要，然后使用公共密钥对它进行加密。

4. Windows NT 使用客户的公共密钥对这个摘要进行解密。

5. Windows NT 将解开的摘要和客户原始发送的随机消息进行对比。

6. 如果两个摘要匹配，这个客户便认证成功。

在这里，你可能会关心与管理员有关的问题，虽然 Windows NT 自身提供了这两种内建的方法来控制服务器的访问，但它并不阻止你创建 ISAPI 过滤器来扩编或替代这种标准的安全机制。我们在第 13 章看到了一些非常基本的过滤器，但你也可以干脆使用其它的方法。你所应该做的就是监视那些由服务器产

生的安全事件，这些事件中最关键的是客户的认证请求。在监视客户认证请求过程中，允许你增加任何必要的措施来确认谁在访问你的站点。

注 ISAPI 过滤器允许你改变 Web 服务器认证用户的方法。

使用数字签名

要精通数字签名的技术是比较困难的，虽然可以做到，但很少有人愿意去尝试。你首先必须懂得的是，数字签名也可以看成是授予证书。你可以把它想象成驾驶执照，因为它们之间有类似的功能。数字签名能标识一些 Internet 对象、以及创建它的人和创建的时间，也能潜在地提供很多其它信息，如果这个对象正好是客户或服务器，数字签名将显示对象的当前拥有者。和驾驶执照一样，数字签名也有过期的时候——它迫使提供商不断地实现它们的承诺，这些过期的数据也使得黑客在篡谋窃取证书上浪费许多无用的时间（因为每个证书都是一个分立的条款，所以黑客只学会如何窃取其中之一是没什么用的）。使用数字签名将帮助人们保持诚实，因为它迫使每个人通过一个中心核查点。在 Internet 上，使用数字签名做期货交易还避免了一大堆的问题：它不依赖于某个人来维护机器的安全性，现在你可直接对那些获得访问的人进行投资（这也意味着要对用户做某些级别的训练，让他们真正学会如何去使用这个特性）。

注 虽然代码符号处理（将在这一节的下载 Internet 部件部分进行讲述）现在还不是数字签名，但最终也将合并到数字签名中来。

实施数字签名是非常直接而简单的，尤其是对客户端。在大多数情况下，商家提供了标准的证书，它能被所有浏览器或服务器所识别，但在授予证书的

实际过程中，它们使用的方式却有不同。例如，在图 14.2 中的 Web 页中便显示了其中的一个潜在的问题，如果你想要获得一个客户的证书，你将选择哪个等级呢？（因为 Class 1 Digital ID 证书 的价格便宜，所以许多人将会首选它）。

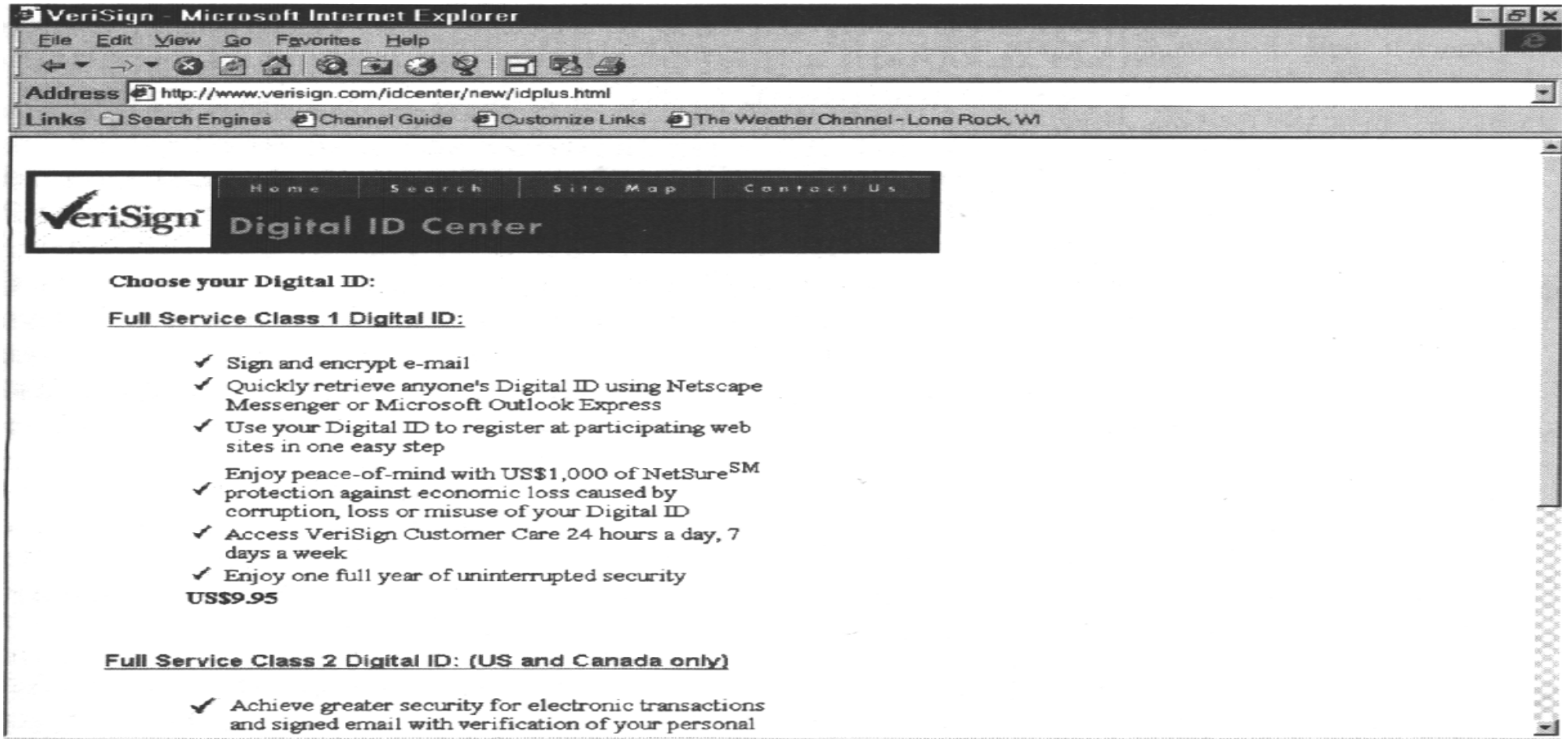


图 14.2　客户端的认证将是未来 Internet 安全性的一个重要部分，但到底选择哪种认证并不容易

从实施的立场来看，你获得了 Class 1 还是 Class 2 证书对你的安全实施并不重要，即便是厂家，也不真正重视它。然而为了兼容性，你可能仍然想坚持用 VeriSign（其实不然）。事实上，不同厂家使用数字签名的方法是非常相似的，因为它们的界面都是由 Web 服务器或浏览器所定义的，你将要做的只是接受一个应用程序，当厂家核查了你的身份后，你将会在 e-mail 中获得一个 PIN 或其它方式的证明，关键是要按照浏览器或 Web 服务器的提示安装这个证书。例如 VeriSign 公司，你进入它的 Web 站点并输入你的 PIN 后，浏览器将自动关注下载和安装的细节。下述段落对 VeriSign 的证书等级作了一个概述。

Class 1 证书，它在一个资源库内提供用户唯一的名字和地址，VeriSign（或你选择的任何其它厂家）将能够核查这个人的姓名和地址。回发 e-mail 是 VeriSign 用来发授证书的唯一手段，要接收这个证书，你必须拥有一个 e-mail 地址，这种方式使得黑客很难伪造这个证书。这种级别的证书每年需花费 9.95 美元（你也可以获得一个 Class 1 证书的免费测试版）。

Class 2 证书，要获得 Class 2 证书，必须由第三方认证你的身份（目前只对美国和加拿大人授予 Class 2 证书）。Class 2 和 Class 1 的最大区别是 VerSign 公司实际通过 EquiFax 公司提供的用户数据库来检测你的信息。你也通过一个硬件签名处理，它需要多个密钥。这种级别的证书每年需花费 19.95 美元。

如果你想获得一个证书并希望它能和多种软件协同工作，那么你不能只看它对浏览器的兼容性，因为数字签名技术有很多的标准规范（请参看表 14.1），

你将发现大部分的数字签名认证厂商已接近这些标准了。

如果你已经有了一个数字签名的证书，并分配给了你的 ActiveX 控件、Web 服务器或浏览器（或以上所有的部件），那么你怎样鉴别其他人有没有证书呢？在你访问不安全的站点时，你总看到一些警告对话框，而访问一个有数字签名证书的站点也与之类似，你将看到一个如下图所示的数字签名证书。

注意，这个对话框还给了你一些优化选项。首先，你要检测证书的有效性，例如，检测它显示的厂商是否是你所期望的那一家；检测其日期来判断是否过期。其次，如果你想要到这个厂商那里去，你应该有能力通过其审计。在证书下的第一个检测框总是允许你在列表中增加一些由 WinVerifyTrust 检测过的指定的公司（我们在第 8 章讲述下载过程中提到过 WinVerifyTrust），如果你选择这个厂商，则每次需要在这个指定的公司下载东西时系统将不再提问，这对访问某些站点来说可能有点冒险，但也许没比这更好方法了，它完全决定于你对这个公司的信任度；第二个检测框允许你对所有授予了指定证书代理权的商家进行访问，如 VeriSign，如果你的认证过程很舒适，你可以不选中这个检测框。

你也许正为要对 ActiveX 控件所做的这些事而感到惊讶。用户和 Web 的主人一样，如果你决定使你的站点公共化，你将发现自己在某些地方需要这些证书。因为人们不愿下载一个未经数字化标识或看不到厂商证书的控件。取得证书的过程应该与我们刚看到的浏览器用户获取证书的过程没有太大的区别。你现在不必立即拥有一个证书来标识你的控件，但在不久的将来，它是必不可少的。我们将在本章有关下载 Internet 部件的部分，介绍标识 ActiveX 控件的实际过程。

理解加密 API

到此为止，防止某些人偷阅你的数据一直是本章所有内容的主题。我们已经学习了一些安全协议和技术，它们能帮助你将黑客拒之门外。然而，只要有

充分的时间，任何一把坚固锁都可能被人撬开。事实上，这就是为何要使用128位的密钥代替40位密钥的原因。由于解开128位的密码“锁”需要太多的时间（至少现在是这样），所以包含在这个记录内的数据对黑客几乎没有吸引力。黑客当然也能打开这把“锁”，但非常困难。

保护层

增加保护层是针对黑客的另一种利器。对数据进行多种级别的加密并加上“防盗门”，黑客要获得数据必须通过这些“门”。只要在你的数据和黑客之间放置足够多的“门”，黑客便很难攻破它。

微软的加密API（或加密应用程序接口）增加了保护层类。它意味着你又可以对敏感数据增加一个保护层了。这里支持的加密技术虽然并非坚不可摧，但黑客要解开它读取你的数据，确实需要花费更多的时间。使用加密API函数将帮助你更好地保护服务器和客户间的数据传输。

Web链接 对于其它涉及Internet的东西来说，加密API（CryptoAPI）是全新的东西，并且还在不断地发展之中。在撰写本书时，加密API只是作为ActiveX SDK的一部分来叙述的，你应该通过Internet下载它的一个详细说明。虽然本书在这一部分提供了最好的叙述，但你还是应该访问一下加密API站点http://www.microsoft.com/security/tech/misf6.htm，以获得最新信息。

加密API还有其它一些用途。因为加密API是作为一个通用工具而设计的，

所以它允许在任何环境下对数据进行加密，而不只限于 Web 服务器。例如，你可以创建一个应用程序将所有存储数据使用相同的加密格式，这些数据可以是存储在本地的、通过 modem 传输的、上载到一个 Web 站点的，或是通过邮件发送的。这将给你的公司带来很多的好处：对敏感数据进行随时加密，使得黑客即便闯进了你的系统，也很难破解你的数据。然而，使用加密技术，也会给你的用户带来一些不便，可以预料，只要加密操作变得更加方便和自动化，你的用户肯定会喜欢上它的。让攻击者的攻击变得困难、摸不着头绪，或者简单地要他们花费更长的攻击时间都是吸引用户的最佳途径。这样麻烦不断的用户就会确信应该在你的系统中保护他们的数据。

注释 Windows NT 4.0 内建了加密 API 1.0（CryptoAPI 1.0）支持。当你安装完服务器修补包 3 后，其支持的等级将升级至 2.0。这两个版本的最大区别是：加密 API 2.0 支持核心加密，它允许开发者在应用程序内纳入加密，而且加密 API 2.0 还支持 X.509 证书、ASN.1 编码、PKCS#7 和 PKCS#10 压缩。你将发现，Visual C++需要有 WINCRYPT.H 和 CRYPT32.DLL 文件，才能提供 CryptoAPI 2.0 的支持。

CryptoAPI 是模块化的，这并不足以为奇，因为在过去的几年里，微软将它的操作设计转移到了模块化的方式上。我们可以拿 CryptoAPI 与 Windows 下的 GDI（图形设备接口）API 相比较，任何厂商都能够增加新的设备驱动程序来告诉 GDI 如何和指定的显示卡工作。CryptoAPI 也是如此，它采用了加密服务提供者（CSP）的思想。虽然 Windows 本身已有微软提供的 CSP，然而，如

果它的加密特性还不能满足你的需求时，你也可以自己设计一个新的 CSP 或从别的公司购买一个。CSP 的安装过程就像是增加一个设备驱动程序一样。事实上，这种设备驱动程序的方式使得在你的机器上可以混合加密硬件和加密软件，这是一个真正的提高点，因为你通常是将硬件和软件作为分立的实体来安装的。

Web 链接 微软已经为 CryptoAPI 开发了一个 DDK（设备驱动程序开发工具），Cryptographic Service Provider Developer's Kit（CSPDK）。你可以通过 http://www.micorsoft.com/devnews/novdec96/crypto5_6.htm 获取更多的信息。

加密一个文件

为了显示一下 CryptoAPI 提供的机能，下面举一个加密文件的例子。加密文件是相当简单的，微软直接依照八个步骤处理（其中六步含有特定的加密函数）。最开始，你象在其它编程中一样，打开一个源文件和目标文件（例子中的这部分代码用 C 语言编写，你也很容易在其它语言中实现），在你拥有一个有效的文件句柄后，你必须像下边的例子一样获得一个 CSP 句柄：

```
//给默认提供者提取句柄
if (!CryptAcquireContext(&hprov,NULL,NULL,PROV_RSA_FULL,0))
{
   MessageBox("Error during CryptAcquireContext",NULL,
 MB_OK |MB_ICONEXCLAMATION);
```

```
    PostQuitMessage(1);
}
```

注释 在 WINCRYPT.H 文件中，你能看到与加密有关的函数、结构和定义的完整列表。

CryptAcquireContext()能够接收五个参数，但在这个例子中只有两个是重要的，第一个参数保存 CSP 的句柄，第二个参数指定你要查找的 CSP 类型。每个 CSP 都有一个名字和类型。例如，当前随 Windows 一起发布的名叫 Microsoft Base Cryptographic Provider v1.0 的类型是 PROV_RSA_FULL。每个提供商的名字是唯一的，而类型却不是唯一的。第二个参数包含了密钥的容器名，如果你在这里输入一个值，Windows 将寻找一个指定的密钥容器，厂商可以在硬件里、注册表中或硬盘里保存一个密钥容器，所以你一般不会知道密钥容器的名字，如果使用 NULL 值（如例子中所示）将告诉 Windows 返回默认的密钥容器；第三个参数包含了 CSP 的名字，你可以通过使用 CryptGetProvParam()函数来取回这个值，填写 NULL 值将返回默认的 CSP；最后的参数包含了一个或多个标志，如果 CSP 没有提供指定的标志值，你可将这个参数设置为 0（微软只为管理方面提供了一些默认的标志值）。

下一步可以是使用一个随机密钥或由口令产生的密钥来加密文件。随机密钥方式在 ActiveX 控件中使用的可能最为频繁，所以在此加以描述，口令方式也与之类似，而实际需要的步骤比随机密钥更少。

```
//创建一个会话的随机密钥
```

```
if(!CryptGenKey(hProv,ENCRYPT_ALGORITHM,CRYPT_EXPORTABLE,&hKEY))
{
    MessageBox("Error during CryptGenKey",NULL,MB_OK |
MB_ICONEXCLAMATION);
    PostQuitMessage(1);
}
```

CryptGenKey()函数提供给你一个唯一的密钥。注意第一个参数是 CSP 句柄，第二个参数包含了加密算法名。微软提供了两种加密算法：CALG_RC2（块密码）和 CALG_RC4（流密码）。算法名会随着厂商的不同而变化，所以知道用户打算使用哪家厂商的 CSP 是很重要的。第三个参数包含了标志，一共有三个默认的标志值，CRYPT_EXPORTABLE 告诉 Windows 能够输出一个随机的密钥到一个 blob（我们将在随后看到它）；CRYPT_CREATE_SALT 告诉 Windows 给随机密钥种子值使用一些其它大于 0 的值；最后，CRYPT_USER_PROTECTED 告诉 Windows 在某个动作发生时提醒用户。最后一个参数是存储返回随机密钥的一个容器。

在获取一个证书的过程中，用户将接收到一个公共密钥，CSP 将这个密钥保存在一个中心位置——用它标识用户。为了保证数据传输的绝对安全，在加密过程中，获取用户公共密钥的一份拷贝来使用是很重要的，这就是下一步所要做的工作。

```
//获得用于交换密钥的公共密钥句柄
if(!CryptGetUserKey(hProv,AT_KEYEXCHANGE,&hXchgkey))
```

```
{
    MessageBox("ErrordwringCryptGetUserKey",NULL,MB_OK |
MB_ICONEXCLAMATION);
    PostQuitMessage(1);
}
```

我们提供给 CryptGetUserKey()的第一个参数是 CSP 的句柄，第二个参数是表示从提供者那里获取什么类型的密钥。每个 CSP 将支持两种密钥：AT_KEYEXCHANGE（这个交换的公共密钥是为应用程序中的个人认证而提供的）或 AT_SIGNATURE（与随后的 ActiveX 控件获取数字签名密钥一样），第三个参数提供存储返回密钥的空间。

我们现在进入了一个有趣的部分。我们获取随机密钥和用户的公共密钥，并将其混合使用，这意味着，即便用户的密钥在某些方面被破译了，数据还仍有另一种方式的加密，它迫使黑客又得重头开始破译。随机因素使得加密技术至少在安全上超过了其它方式的加密。

```
//决定键 blob 的大小并分配内存
if(!CryptExportKey(hKey,hXchgKey,SIMPLEBLOB,0,NULL,&dwKeyBlobLen))
{
    MessageBox("Error computing blob size",NULL,MB_OK |
MB_ICONEXCLAMATION);
    PostQuitMessage(1);
}
```

```
if ((pbKeyBlob = malloc((dwKeyBlobLen)) == NULL)
{
    MessageBox("Error allocating memory for blob ",NULL,MB_OK |
MB_ICONEXCLAMATION);
    PostQuitMessage(1);
}

//将会话的密钥输出到一个简单的密钥 blob 中
if(!CryptExportKey(hKey,hXchgKey,SIMPLEBLOB,0,pbKeyBlob,&dwKeyBlobLen))
{
    MessageBox("Error during CryptExportKey",NULL,MB_OK |
MB_ICONEXCLAMATION);
    PostQuitMessage(1);
}
```

注意输出 Blob 密钥（用户的公共密钥和随机密钥的组合）的过程，实际上有三步处理，我们调用了 CryptExportKey()函数两次，头两个参数值是我们创建的密钥：随机密钥和用户密钥；第三个参数告诉 Windows 创建什么类型的 blob，大部分 CSP 支持两种值：SIMPLEBLOB 或 PUBLICKEYBLOB；第四个参数是一些标志，除非 CSP 为别的目的而使用它，否则这个参数总设置为 0；第五个参数是指向存储 blob 的缓冲区的指针，如果你在调用 CryptExportKey()时将它

设置为 NULL，它将简单返回缓冲区的大小，这个大小在第六个参数中将用到，这就是我们在第一次调用它的目的；第六个参数一般包含了你实际调用 CryptExportKey()创建 blob 时所需缓冲区的大小。（这里的源代码也包含了一个内存分配函数的调用）

既然我们已经获取了一个 blob，那么现在该关注一下后续工作了，你马上要做的工作是销毁用户的公共密钥，因为黑客可以从你的内存中搜寻到你的密钥并破坏你的代码，所以从内存中将密钥删除，无论从内存的角度还是从编码练习的角度来看，这都是一个很好的策略，它对维护你的安全系统来说，是必不可少的。下面的代码是调用 CryptDestroyKey()函数来删除公共密钥。

```
//释放键交换密钥的句柄
CryptDestroyKey(hXchgKey0;
hXchgKey = 0;
```

在这时，你可以将 blob 数据保存到磁盘或硬盘内的一个文件中。blob 实际上组织了一个头，接收的机器将使用它对文件进行解密。然而，一个没有数据的头并不好，这就是我们下一步所关注的，你要做的第一件事就是定义数据块的大小，如果你使用块加密方式，CSP 将提供块大小，块加密一般需要你在文件的尾部增加一个附加的空白块。当使用流加密时，你可以定义任何大小的加密块（尽管微软推荐使用 1,000 字节的块大小）。现在你将数据一块一块读取到缓存，并使用以下代码对它进行加密，然后将数据块写入目标文件中。

```
//对数据进行加密
if (!CryptEncrypt(hKey,0,eof,0,pbBuffer,&dwCount,dwBufferLen))
```

```
{
    MessageBox("Error during
CryptEncrypt",NULL,MB_OK|MB_ICONEXCLAMATION);
    PostQuitMessage(1);
}
```

你可以设计一个循环来不断地处理数据，直到读完数据块为止。确认你填充了所有不满的块，CryptoAPI 甚至将块大小作为加密处理的一部分。

14.3 蛮荒不化的互联网

一些人把 Internet 看作是计算机的现代野蛮西部，因为在 Internet 上实施的标准实在太少，几乎没有什么安全手段。虽然在 Internet 上出售商品的人正以指数方式增长，但不幸的是，蛮荒不化的 Internet 环境使许多慎重的行业，如银行和大型零售商场都不敢问津。造成这种情况的原因很简单：它们不愿意承受可能出现安全漏洞的风险。所以，就在许多新兴的公司，如 CD-Now 正高兴地迎接 Internet 的同时，另一些需要 Internet 的老牌公司却对它持观望的态度。

虽然 Internet 技术在不断成长革新，但它还是招致了“现代野蛮西部”的“雅称”。Internet 是内部技术、新技术和老技术的结合体，然而，这正是威胁 Internet 的最大原因。例如，当前版本的 IP 协议（互联网协议）不支持任何方式的加密数据（至少在协议级没有支持），问题是，如果人们想通过加密的形式在 Internet 上传送数据，将没有工具来确认其安全性是完全可靠的。新版本的 IP 将解决这

个问题，但现在还尚未出台，随着时代的发展，Internet 必定会在这方面取得成功。现在虽然有那么多这样或那样的升级承诺，但是很少有实现的。对于某些人来说，在 Internet 上“冲浪”是非常有趣的，因为它是那么自由自在、无拘无束。

Web 链接 通过浏览 http://www.genome.wi.mit.edu/www/faqs/www-security-faq.html，你可以看到在运行 Web 服务器时所要担负的一些安全风险。它包括了使用某种浏览器或服务器时所冒的风险，其中包含了当前的 Windows NT、Unix 和 Macintosh 服务器，最后它倾向于使用 Unix 服务器。该站点也给出了许多通用问题的答案，比如，如何在你的严格监视和保护用户的隐私之间找到一个平衡，它甚至还告诉你，文档文件是如何随着 Internet 的发展而变化的。

就像野蛮的美国西部最终发展成了现代文明国度一样，Internet 也将最终走向辉煌。类似 ActiveX 控件这样的技术将使 Internet 最终变成一个安全的商务场所，这些技术也将把 Internet 变得更加友好；就像汽车和高速公路把美国西部变得更加容易旅行一样，这些技术也将加快人们从一个地方到另一个地方的交流速度。

我们必须要解决的一个难题就是安全问题。只有在一个安全的环境内，人们才能够安心高效地工作，否则，人们将在与敌人的死死纠缠中花费太多的时间和精力。下述各节将讨论隐藏在 Internet 安全问题背后的相关课题。你在应用程序内处理安全问题之前，应该先弄清楚有什么样的安全问题，如果你现在已经明确了所有安全问题，并想实施安全策略时，请看看讲述 Internet 安全标准的

那一节，否则，如果你想把 Internet 作为一个工具应用于你的公司，你应该先花点时间来思考一下将要面临的一些安全问题。

从商业角度看 Internet

就像前面提到的，许多人为毫无限制的 Internet 感到不安，你确实可以在 Internet 上看到几乎任何东西，因为没人对它进行规划（美国政府最近对 Internet 在线服务的规定并没有奏效）。如果你是一个编程人员，Internet 这个无拘无束的环境也许还有某种魅力，但你不是在为自己编程，你必须考虑到商业人员的需求，商业人员当然不希望花了钱却买回了不安，四面漏风的工作环境对他们来说就像是一场噩梦，他们希望每件事都井然有序，有条不紊，最重要的是要安全可靠。

急速膨胀的 Internet 并没有对商业用户注入多少信心。在一些商业用户的眼里，新的和未经考验的技术只会在茫茫的等待中发生。考虑一下 ActiveX 控件的潜在得失，它确实加快了商业用户的办事速度，但还是有一些安全因素值得担忧，因为任何一个具有少量编程经验的人就可以创建一个潜伏破坏性的控件，并把公司的一切搞得一团糟。

最近的商业调查报告显示，大公司在 Internet 上变得审慎了，他们采用审慎方案的一个最大原因是缺乏安全感，这些公司希望能证实技术的可靠性，并保证它们的数据在任何通信中都很安全，建立这种值得信赖的 Internet 的一种方法是在内部编写优良的应用程序，也就是说，创建 ActiveX 控件。

注释 商业用户需要处理的主题很多，它们一般与编程人员无关，例如，是否允许职员访问 Internet 以及允许进行什么层次的访问？这些当然是个热门的话题，但我们在这里不做全部的描述，我们只是讲述一些与编程人员相关的主题。

那么你该如何处理这些安全问题呢？你可以与公司的主管交谈，如果你发现了他们最担心的只是你编写的应用程序，那么你应该说服他们，作为应用的一部分，安全也是非常重要的。例如，我们可以使用一些方法来访问 Windows NT 服务器内部提供的安全性，这种方法的缺点是访问安全特性将需要许多附加的代码，使得应用程序的运行变慢，并使用户也更辛苦，你也应该在程序的设计阶段告诉主管对其所进行的权衡（应用程序最终将在 Web 站点上应用）。

注 要实现公司各阶层间的通信，必须建立一个安全的 Internet 或 Intranet 环境。

定义要保护的对象：数据

那么，我们需要保护的到底是什么东西呢？你也许会得到许多答案，但它们实质上都是指向一个东西，所有说的和做的都是为了保护数据的安全。数据所牵连的影响甚至超过了硬件和软件，硬盘丢失了，你还可以替换它；软件配置破坏了，你还可以重新安装它，可是公司的财经周报丢了，你就得重新处理它了。数据是计算机环境中最精确的部分，因为它一旦丢失，便很难再修复。

保护数据的安全是所有人在使用中最应该关心的事。很多网络管理员在照

顾安全事务上花费的时间最多，在大多数情况下，安全性关系到了谁做了访问、访问了什么数据以及访问的原因等。随便翻翻任何一家商业出版物，你都能看到一篇又一篇的文章讲述数据的安全性。我们每时每刻都在用计算机处理数据，这些数据对你公司的重要性是毋庸置疑的，当你创建一个 Active 控件，并连接到 HTML 页面上，此时你最大的安全顾虑就是你将要提供的数据。

数据安全涉及到了访问的问题，不管是访问本地机器还是访问网络，有一个目标是相同的，那就是防止非授权的访问。在本地机器建立一个安全的环境是相当容易的，除了有各种软件技术外，许多计算机还允许 BIOS 层次的口令保护，如果这还不够，你还能依靠物理的安全保护来保证别人访问不了你的系统。然而，要在网络上实施这些安全性就比较困难了，但你还是不用花太多的努力就可以做到，流行的网络操作系统产品如 Novell 的 NetWare 和微软的 Windows NT，都提供了各种安全手段来保护本地数据（除非有人在网络的物理电缆上安装上了网络监听器）。在广域网上（WAN）实施安全性简直就是做恶梦，尤其是当你还有许多拨号上网的机器，不管连接的距离如何，由于它们使得软件和硬件的项目变得更丰富了，所以某些人便更容易偷偷闯进你的网络。有些人甚至断言，在 Internet 上实施安全性是不可能的事。不只是广域网带来了这个重大的问题，Internet 自身提供的公共访问也带来了一些问题。任何人，只要看过最近的商业杂志便知道那些软件公司是如何倾尽全力地来修补他们的产品中的安全漏洞的。

建立某些形式的保护

即便你现在尽量忽视Internet方面的安全问题，数据保护还是有许多其它方面的主题。例如，你将提供什么类型的数据保护？让我们暂时从非Internet的角度来探讨一下，在这里，你能够从硬件或软件的角度来看待安全保护。一些公司为了保护那些需要特殊照顾的数据，他们把工作站从其它项目中分离出来，与网络的物理分离意味着，任何人如果想要访问机器内部的数据，必须实际地使用这个工作站，必要的话，你还可以通过使用加锁来防止别人访问它。有的公司还通过使用数据加密或执行安全审计来解决这个问题，这些都是用软件实现的安全策略。

不幸的是，在Internet上，你不可能把自己的机器锁在房间内来达到保护数据的目的，这意味着，随着Internet的发展，过去使用硬件方式保护数据的方法将会被淘汰。硬件级保护的失去增加了编程人员的负担。作为一个编程人员，你必须负责在应用软件中增加一些安全手段（包括一些产品如ActiveX控件，ISAPI filters/extensions）。幸运的是，你能够参考许多新的安全方法，例如，文件加密已经是一个实用的保护技术。无论在广域网上还是Internet站点上，加密文档的安全特性都一样，只是允许访问的人数不一样。

我们已经讲述了工作站的一些安全问题，让我们再来关注一下服务器。从Internet的角度来看，有些公司把自己的站点和网络从物理上完全脱离开来，（这限制了你将这个站点用于公司内部需求的方法）。使用一个分离的Web服务器是保护你网络安全性的一种物理方法，没有人能够通过这个Web服务器来窥探你网络内部的数据。防火墙是一种软件的保护方式，这种方式的登录过程，实

质上在局域网上已使用过多年了。

你当前使用的或以后将要实施的计划都有一些实际的极限，其中，你最应该考虑的是 Internet 访问的发展将越来越灵活。当然，你可以关闭你的站点来防止任何人来访问它，但你此时又能做什么呢？采取顽固和粗糙的策略来实施你的安全性可能不会有太大的效果。为了让人在 Web 站点上学习加强安全性的新方法，而不分配给他们访问的级别，将是你面临的一个挑战，从编程者的角度来看，这意味着你应该学习安装在网络上的硬件和软件的应用能力，已避免陷入 Internet 的安全顾虑中。

实施方案

要创建一个安全网络环境是一个庞大的项目，即便是系统地陈述一个基本的安全方案，你也要花费许多时间和精力，而要具体实施它则需要更长的时间，尤其是为了减少对工作环境的冲击，而使用阶段性的实施方案时，时间就更长了。即便你制定了一个非常安全的计划并尽己所能地考虑到了所有的意外，你也不能完全确信已经面面俱到了。当最后有人闯进了你的安全系统并破坏了一些数据时，你便只能打道回府了。

一旦考虑到所有这些潜在的问题时，你不难得出为什么网络管理员要经常地对用户的权限进行分配，哪怕他是在做一件无关痛痒的事。例如，一些网络管理员强烈地反对公司进行任何形式的 Internet 访问；而另外一些网络管理员则一旦证实有 Internet 访问，便要像“老大哥”一样监视用户的活动情况。

不管建设什么形式的 Web 站点，你面临的将不止是网络管理和数据保护的

问题，考虑用户的提问也很重要。如果有人告诉你连接是安全的，而你发现它并非如此，你又怎会喜欢它呢？许多信用卡用户真不愿意这样冒险，从计算机到 Web 站点的一个不安全的连接可能意味着信用卡内数千元的损失。

在实施一个安全方案之前，你应该已经完成了这些工作：确认你创建的应用程序能够长时间保证客户和服务器间数据交换的安全；如果你想把你的 Web 站点作成防弹（bulletproof）形式，你应该考虑到所有使用 Internet 人的需求，以及可能产生的冲突。当一些没有考虑到的事情发生、并且确实有人闯进了你的网络时，你便能够立即制定一个安全计划来修补这些网络安全漏洞。

14.4 确保 Internet 下载代码的安全

许多公司正承诺把它们的应用软件当作一系列 ActiveX 控件上载到 Internet 上，例如，Lotus 计划将一系列 Notes_specific 阅读器以 ActiveX 控件方式迁移至 Internet 上，Quarterdeck 公司也计划把它们的整个应用系统程序转换成 ActiveX 控件形式并上载到 Internet 上。你很快就能够直接通过浏览器查看各种类型的数据，在某些情况下，你还可以直接对它进行编辑。（要得到它们成熟的产品，我们也许还得等待一段时间）。

如果商家没有计划对下载的代码进行检验、用户在不能确认代码是否受到过病毒的侵害时，就盲目将其保存，将会带来很大的危害。我们已经在第 8 章中阐述了 Windows Verify Trust API（用于连接 Windows Trust Provider Service）背后的一些技术，在这一节中，我们将关注这些 API，你将对这些技术的运作

方式有一个总体的了解。

在这一节中，我们也将关注一下 Windows Software Publishing Trust Provider，它实际上是 Windows Verify Trust API 的一个附加系统，它用来检验下载的软件部件是否值得信赖，其检验的方法有多种，其中包括检验本地的规则（如在浏览器上与安全相关的检查框）和检验文件自身的加密信息（如数字签名）。

最后，我们将关注对 ActiveX 控件签名的过程。将签名过的控件放置到 Web 服务器上，将使用户在下载它时，不会再提示“not trusted”（不能信赖）的信息。这部分也将让你学会如何为测试而清理自己的机器，你将学会如何卸载一个 ActiveX 控件，这样，你不必在每次测试控件的签名以及在你的 Web 站点下载其它特性时，都得重新清理机器。

使用 Windows Verify Trust API

这个特殊的 API 在某些方面现在已经相当稳定，微软已经在 IE3.x/4.x 以及 Windows 自身中部分地实施了它。你阅读到的其它部分正处于不断的改变中，因为很多软件公司正在各自地研究检验下载软件的最好方法。

那么，Windows Verify Trust API 的精确定义是什么？它是决定你是否能信任任何 Windows 对象的一种通用方法，这些对象可以是客户请求的服务、服务器请求的信息、下载的文档文件或甚至是 ActiveX 控件。这个 API 的最终形式是允许你检测任何对象的可信度。

就像 Windows 支持的大多数 API 一样，Windows Verify Trust API 也是可以

扩展的，你可以增加新的特性来允许它执行一些扩展的检测。其中的一个扩展是 IE 3.x/4.x 带来的 Windows Software Publishing Trust Provider，我们将在下一节进行阐述。现在，你所要知道的一点是，Windows Verify Trust API 只是个通用的 API，随着更多人对它的使用，它可能需要更多的扩展。

Windows Verify Trust API 使用多种方法来检测文件的可信度。其中的一些方法尚存在争议，但有两种方法最为通用，即检测系统规则和校对伴随于对象的证书或数字签名。你将发现 Windows Verify Trust API 也依赖于外部的证书，例如，当前许多流行的 Internet 加密标准正使用公共密钥和私有密钥方式，公共密钥驻留在文件头上，私有密钥存在于用户的机器内。要破解一个加密文件，你必须要既有公共密钥又有私有密钥，因为只有用户拥有这个密钥，所以其他人将不能阅读这个文件。显然，还有一些比两把密钥更复杂的方法，有些简单的方案还增加了随机密钥，他与公共密钥相结合，使得黑客几乎没有可能破解它。

系统规则存在于许多地方，例如，浏览器的“信任”信息存储在配置文件或注册表中。系统管理员也可设置一些策略，这些设置可以在一个单独的用户注册文件中或在所有用户都可使用的通用注册文件中。策略所在的精确位置依赖于你使用的 Windows 版本，和是否打开了多用户配置（在 Windows 95/98 中），以及系统管理员实施的策略类型（系统的或单独的）。你也可以使用委托供应商（CPS）规则（准确的名称叫做“trust provider”，因为其中的规则来源可以是任何委托代理，而不止是 CPS），例如，你可以告诉浏览器，任何经过某个委托供应商认证的人都是可信的。作为受托检验规则，委托供应商提供了指定

的拥有者的列表，并定义了指定对象的类型和委托供应商的级别。用户的操作也能够影响 Windows Trust Verification API 的使用规则，例如，如果用户告诉系统，某个供应商是可信的，则这个信息便存储于注册表中，而 Windows 的受托检验服务每次在证书中看到这个商家的名字时，便不再对这个对象作深入的检测。

现在，你已经对这个 API 的功能做了一个概要的了解，让我们再关注一下这个 API的自身。你将会对下面这个函数感兴趣：

```
WinVerifyTrust(HWND hwnd,DWORD dwTrustProvider,DWORD dwActionID,LPVOID
ActionData);
```

正如你所看到的，这个函数需要四个参数，大部分人应该牢记第一个参数，它是当前窗体的句柄，这个参数的目的是让 WinVerifyTrust()函数知道当前正有用户在做决定，例如，函数可能想询问你是否下载一个没有签名的文件。如果想要在不干扰用户的情况下检验一个对象的可信度，你可以简单使用 INVALID_HANDLE_VALUE 来代替窗体的句柄；如果你想要用户的桌面代替当前的应用程序，对任何交互作用都做出反映，你也可以将参数值设为 0。第二个参数定义了对谁来询问与信任相关的事情，Windows 认识两个默认值（尽管商家也能定义任何指定的值用来明确真正的委托供应商）：WIN_TRUST_PROVIDER_UNKNOWN（基于你想要执行的操作，找到一个代理商）或 WIN_TRUST_SOFTWARE_PUBLISHER（一个实际的软件发布者）。如果你选择 WIN_TRUST_PROVIDER 选项，Windows 将尽量找到一个包含你要执行操作的注册项。如果不能发现这样一个注册项，WinVerifyTrust()函数将

返回一个 TRUST_E_PROVIDER_UNKNOWN 的值。第三个参数指定了一个操作，它告诉代理供应商，你想要做的操作，因为每个代理供应商都不同，所以你将必须检查代理商提供的文档，最后一个参数的准确内容也依赖于你使用的代理商。在大多数情况下，你将至少需要告诉代理商你想要检验的数据，有些代理商也可能要求有关信任级别的信息或者信任裁决的内容。

WinVerifyTrust()函数在调用执行后，一般返回代理商的特定值；在某些情况下，它也返回四个标准值中的一个，你将注意到这四个标准值全是错误信息（Windows Verify Trust API没有定义任何默认的成功信息），下面就是这四个值的解释。

TRUST_E_SUBJECT_NOT_TRUSTED　通常，代理商会提供给你更多的错误信息，这个返回值简单地提示：检验的对象没有通过你指定的检验操作，它不代表这个对象从根本上不值得信任，而只是不能被当前指定的操作信任。除非代理商支持某些类型的通用操作，或者你只需要对这个对象执行单一的操作，否则你将需要为每个操作分别调用WinVerifyTrust()函数。

TRUST_E_PROVIDER_UNKNOWN　就像前面提示的，Windows基于当前的定义操作，不能找到指定的代理商时，便返回这个错误信息。

TRUST_E_ACTION_UNKNOWN　如果你要求的操作不被指定的代理商支持时，函数将返回这个错误值。WinVerifyTrust使用注册项来检验有效操作，而不是真正与代理商进行交流，这意味着，即便代理商支持指定的操作，你也可能会因为注册被损坏而不能执行某个操作。

TRUST_E_SUBJECT_FORM_UNKNOWN 产生这个错误信息的原因是多方面的，多数情况下是因为参数不正确或参数中包含了不完整的信息。如果代理商不能找到你想要检验的对象，也将返回这个信息，幸运的话，代理商将提供与数据相关的更详细的信息，但如果代理商不能判断问题的来源时，你或许也只能得到这个值。

理解 Windows 软件发布信任供应商

Windows 软件发布信任供应商（Windows Software Publishing Trust Provider）附加于 Windows Trust Verification API（我们在前一节中已进行了阐述）之上，这个附加系统的主要目的是允许应用程序检验软件部件中是否包含数字签名或证书，其中的每个项目都将检验用户本地系统中的文档，并判断它们是否是由委托的供应商作为权威的软件发行的。对于 Windows Trust Verification API，这个 API 使用丰富的技术和信息资源来判断指定的文档是否值得信赖。

你将发现 Windows 软件发布信任供应商也使用 WinVerifyTrust()函数，在前几节中我们讨论过这个函数，但这里的使用略有一些不同，首先，你总应使用 WIN_TRUST_SOFTWARE_PUBLISHER 委托供应商，除非委托供应商支持别的值，如果使用 WIN_TRUST_PROVIDER_UNKNOWN 委托供应商，Windows 将简单选择默认的委托供应商，Windows 也定义了两种操作（你可以提供其它可供选择的委托供应商）：WIN_SPUB_ACTION_TRUSTED_PUBLISHER（检验文档的供应商是否在信任列表中）以及

WIN_SPUB_ACTION_PUBLISHED_SOFTWARE（检验文档自身是否有正确的鉴定证书）。当前版本的 Windows 软件发布信任供应商还不支持 WIN_SPUB_ACTION_ TRUSTED_PUBLISHER 操作，如果选择 WIN_SPUB_ACTION _PUBLISHED_SOFTWARE 操作，WinVerifyTrust()也将期待 WIN_TRUST_ACTDATA_ SUBJECT_ONLY 数据结构，该结构如下所示：

```
typedef LPVOID WIN_TRUST_SUBJECT
typedef struct WIN_TRUST_ACTDATA_SUBJECT_ONLY
{
    DWORD dwSubjectType;
    WIN_TRUST_SUBJECT  Subject;
}WIN_TRUST_ACTDATA_SUBJECT_ONLY,
      *LPWIN_TRUST_ACTDATA_SUBJECT_ONLY
```

请注意，这个结构包含了两个变量，dwSubjectType 定义了你将要检验的对象类型，你可以为大部分数据文件选择 WIN_TRUST_SUBJTYPE_RAW_FILE，或为可执行文件选择 WIN_TRUST_SUBJTYPE_IMAGE（其中包括 DLL 和 OCXs），Subject 结构指向你要检验的对象，其格式为：

```
typedef struct _WIN_TRUST_SUBJECT_FILE
{
    HANDLE  hFile;
    LPSTR  lpPath;
}WIN_TRUST_SUBJECT_FILE,*LPWIN_TRUST_SUBJECT_FILE;
```

正如你所看到的，这个结构中的两个变量分别指向文件和文件的路径。在大多数情况下，你将在浏览器或其它应用程序的缓冲文件夹内找到这个文件，例如，Internet Explorer 3.0 将它的缓冲内容保存在 Internet 临时文件中（数据文件）或 OCCACHE（可执行文件）文件夹中，它们位于 Windows 的主目录下，而 Netscape Navigator 的大部分 ActiveX 控件将临时数据保存在 ActiveX 控件缓冲文件夹（也在 Windows 的主目录下）。

那么 Windows 到底是如何检验文件呢？WinVerifyTrust()在执行中，首先寻找 PKCS#7 标志的数据结构，它是在为控件签名过程中创建的（我们在下一节中将要讲述）；然后再寻找一系列 X.509 证书，在当前实施的 Windows 软件发布信任供应商中，你必须伴随着软件发布的公开密钥，再提供一个私有密钥（我们将在下一节中讲到）。在未来，Windows 软件发布信任供应商也将寻找 X509.3 扩展定义的密钥使用限制和其它属性，如果检验到 PKCS#7 数据结构和 X.509 证书都是正确的，WinVerifyTrust 将给你的应用程序返回一个成功的信息。

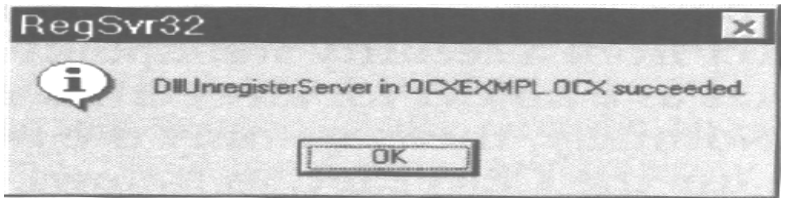
Internet 部件下载机制

你也可以把本节命名为“对代码进行签名”，这个过程实际上包括四步，你可能要多次重复这些步骤才能得到正确的结果。第一步是创建一个在本地 Web 服务器上使用 ActiveX 控件的文档，你把这个服务器当作一个 Internet 站点来访问，我们在第 10 章中已经讨论了它的详细过程，这里就不再介绍了；第二步是删除控件的注册信息，许多编程语言为你创建的控件自动注册，即便不能自动注册，你也要作为测试的一部分来注册它，去掉控件注册后，所有的注册项被

删除，这些项是 WinVerifyTrust()用来寻找控件的一个本地拷贝时使用的，当它找不到注册项时，你所看到的屏幕将与用户看到的一样；第三步是对你的 ActiveX 控件进行签名；第四步也是最后一步，就是访问用来测试的 Web 站点并查看 Web 页，它将下载这个控件并允许你测试签名的过程，你可能需要对后三步进行多次测试才能最终确认签名过程的正确性。

去掉控件的注册信息

在 SYSTEM（或 SYSTEM32）目录下，有一个叫 RegSvr32 的小程序，它用来负责增加 ActiveX 控件在注册表中的登记项，使用命令行开关-U，它也可以取消控件相应的注册项，如果你想要取消第 10 章中创建的基本控件的注册项，那么你可以在命令行输入 REGSVR32 OCXEXMPL.OCX -U。成功删除注册项时，你会看到如下图所示的对话框。



签名你的控件

对你的控件进行签名也分为四个步骤，第一步是创建一个 X.509 证书（我们在前几节中从安全的角度讲述了这个证书），你将使用 MAKECERT 应用软件来创建一个基本的证书。这个应用软件的文档在 SIGNCODE.TXT 文件中，

你可以在 ActiveX SDK 的安装软件的 BIN 目录下找到这个文件。

注释 Visual C++5.0 提供了 MAKECERT 这个应用软件，你将需要它来签名你的控件，然而，在安装中并不加载这个应用软件，在 Visual C++ 安装光盘的 Cab&Sign 目录下，你将找到它的所有文件，确认你已经把这些文件拷贝到了硬盘中（它不能在光盘中运行）。以下处理过程假设你使用 ActiveX SDK 获得了最新的工具，但这里使用的是 Visual C++5.0 提供的工具。

```
MAKECERT-U:AKey-K:AKey.PVK -n:CN=MyCompany -d:A-Company TESTCERT.CER
```

这个命令行是用来创建一个随机的公开/私有的密钥对，并将这个密钥对和便于识别的名字联系起来。我们也创建了一个私有密钥文件（PVK），它保存私有密钥的备份，你创建好一个证书后，必须将它和一个基础 X.509 证书结合起来，这个证书来自诸如 VeriSign 的 CSP。由于现在还没有任何 CSP 来处理这类证书，微软提供了一个测试证书，你可以通过调用 ROOT.CER 来使用它，它和其它文件都在 BIN 目录下。

我们将使用 CERT2SPC（certificate to Software Publishing Certificate）应用程序来将两个证书加入到 PKCS#7 签名块中，这个对象实质上充当包括在签名对象中的证书的句柄，一般只有两种证书，以下我们使用其中的 CERT2SPC。

```
CERT2SPC ROOT.CER TESTCERT.CER TESTCERT.SPC
```

在这个例子中的三个参数分别是微软提供的基础证书、我们在第一步中创

建的证书以及用来存储 PKCS#7 签名块对象的新文件名。现在你已经有了一个完整的证书，在第三步中，你将把它加入到 Active 控件中，即使用以下命令行。

```
SIGNCODE -prog OCXEXMPL.OCX -spc TESTCERT.SPC -pvk MYKEY.PVK
```

这个命令行实际完成了很多任务，它做的第一件事是创建一个加密摘要的映像文件（在此例中是 OCXEXMPL.OCX），然后用私有的密钥标志它，这个加密的摘要允许客户对映像文件的当前状态和它接收到的文件状态进行比较，客户将使用这个比较结果来检测文件是否有任何形式的损害；下一件事，SIGNCODE 删除从 SPC 文件中发现的所有 X.509 证书，并使用 X.509 证书的序列号和已标志的加密摘要创建一个新的 PKCS#7 数字签名对象；最后，它将这个新的 PKCS#7 数字签名对象随同 X.509 证书一同嵌入到映像文件中。

考虑到 SIGNCODE 程序所做的大量工作，你将需要检验它的结果，这就是第四步要处理的工作。首先要检验数字签名是否正确地嵌入了文件，你可以使用 PESIGMGR 应用程序来处理，你所要提供的只是被检验文件的名字，如果获得一个成功的信息，便可使用 CHKTRUST 程序来检验可执行代码是否已用数字签名正确地表达了，这个程序与用户浏览器所做的工作相似，它首先检验数字签名对象，然后是 X.509 证书，如果两者都检查完毕，CHKTRUST 也将检验映像文件的代码，确认是否与数字加密摘要相一致。CHKTRUST 最后所做的处理是浏览器一般不做的，那就是检验多种证书之间的连接关系并确认独立的链最后终结在基础证书位置。

完成实际测试

检验已签名控件所要做的最后一步是实际使用该控件。你必须通过 Internet 连接来进行测试。简单地将控件移入你的 Internet 服务器中，确认它不再在你的本机上有注册，然后尽量使用浏览器来查看测试页。如果一切工作正常，当你下载这个控件时，你应该看到一个显示出来的证书，而不是一般的警告信息。

即便控件被正确地签名了，也还有一些东西会导致你的测试失败。可能发生的第一个错误是没有正确格式化<OBJECT>标志，这个标志在第 8 章中我们已经讨论过。请确认你的 CODEBASE 属性指向了控件所在的位置，另外，在你的机器上使用 URL 来代替目录位置，例如，如果控件位于 Web 服务器的 CONTROLS 子目录下，则 CODEBASE 属性使用如下的 URL：http://www.*mycompany*.com/CONTROLS/OCXEXMPL.OCX。

在本机做好签名后，你可能想双重检验你的控件。虽然控件在签名过程中受到损害的情况很少，但还是存在这种可能性的，如果你第一次在远程位置测试时，控件工作不正常，那么请确认控件在你的本地测试页上工作是否正常。

14.5 安全标准

在前几节中，我们集中介绍了为 Internet 编写代码将要面临的所有问题，这些问题确实存在，但你没有必要一个个地处理它们。在你阅读本书时，标准化组织正巧提出了保护数据的方法。你所要做的就是学习由这些组织提出的管理 Web 站点的方法。采用这些方法有两方面的好处：首先，你不必重新摸索和创

造这些标准；其次，你所采用的安全方法能够与其它站点采用的方法相一致，它减少了用户在学习中所走的弯路，可以使其他编程者使用那些已开发好的工具。

Web 链接 如果你想寻找有关 Internet 安全标准的最新信息，请访问 http://www.w3.org/pub/www/security/，这个 Web 页虽然只提供了概括性信息，但它能让你连接到其它站点上以获得更多的信息；另一个好的站点是 http://ww-ns.rutgers.edu/ww-security/reference.html，它放置了 Rutgers 大学网络服务中心提供的 WWW 安全会议资料，这个站点主要告诉你各种安全会议的联系方法以及摘要信息，而不是有关安全的详细规范；如果开发者想要获得商业角度的安全信息，可以访问 http://www.rsa.com/，RSA 站点覆盖了一个非常广阔的主题，其中包括 MasterCard 和 VISA 开发安全信用卡事务的当前状况；通过 ftp://ftp.isi.edu/internet-drafts/lid-abstracts.txt，你也能看到 IETF 产品的当前状况。

下面，让我们看一看 Internet 的各类安全标准，这些标准有的已经实现，有的正在成为标准。表 14.1 显示的是在编写本书时已有的标准或标准的草稿，在你阅读本书时，又会出现更多的标准，令人惊奇的是，在 Internet 上的厂商标准可能是除浏览器技术外增长最快的一个领域，（浏览器技术的发展速度如此之快，以至于测试者都难以跟得上它）。你还将注意到，这里列出的主要标准不是来自微软或其它公司，而是来自下述两个组织之一：IETF（Internet Engineering Task Force，Internet 工程任务组）或著名组织 W3C（World Wide Web

Consortium）。IETF 已有很长的历史，它是最早为 Internet 工作的组织。W3C 对每一类要用于 Internet 的新标准负责，例如，微软现在正努力征得 W3C 对其<OBJIECT>标志和其它与 ActiveX 相关的 HTML 扩展的许可。

表 14.1 Internet 当前的安全标准

标准	描述
分布式身份认证安全服务（DASS）IETF RFC1507	DASS 是 IETF 正在改进中的一个标准，它为在 Internet 上提供身份认证服务定义了一个试验的方法。这里认证的目的是确认发送消息或提出请求的人是谁。DASS 正在努力解决当前保密字设计中存在的许多问题，例如，现在还不能检验出发送保密字的人是否在模仿他人。由于 DASS 是在一个分布式环境中提供认证服务，所以它有着特殊的挑战，因为用户不止是登录一个机器，他们能在网络上登录所有能够想到的机器

续表

DSI（数字签名优先权）	这个标准由 W3C 制定，它用于克服通道级安全的一些限制。例如，通道级安全不能对文档或应用程序的语意做处理；因为所有的处理过程都发生在 Internet 上，而不是在客户或服务器端，所以通道也不能有效利用 Internet 的带宽。DSI 为传输签名定义了一个数学的方法，它实质上是对指定的个体或公司做唯一的表述。DSI 也为标识安全属性提供了一种新的方法（PICS2），为维护提供了一种新的格式（PEP）。DSI 标准也被建立在 PKCS7 和 X509.V3 标准中
扩展的 Internet 标记 SHTTP（EIT SHTTP）	它是 SHTTP 的扩展，它在当前的 HTTP 列表中增加了与安全相关的标志。目前还没有标准化组织支持这种技术，但以后将会有的。（IETF 最近成立的 Web 事务安全组织正关注着这一技术，你可以通过 http://www-ns.rutgers.edu/www-security/wts-wg.html 来进一步了解详细信息）

续表

通用安全服务应用程序接口（GSS-API）IETF RFC1508	这是个已批准的 IETF 规范，它定义了以通用方式支持安全服务调用的方法。使用通用接口允许在更广泛的平台提供更多的源代码级可移植能力。IETF 并没有把这个标准看成是这类标准的结束，而是一个开始，将来会有更多的规范和标准。然而，学会这些已有的标准将帮助你在各种安全实施方法中找到通用的线索
通用安全服务应用程序接口（GSS-API）C-binding RFC1508	这是个已批准的 IETF 规范，它定义了 C 语言中使用支持服务调用的方法，这个标准是最先基于 RFC1508 实现的标准之一
Internet 协议安全协议（IPSec）	IETF 最近成立了一个 IP 安全协议工作组来寻找 IP 安全性方面的问题，比如，在 IP 协议级无法对数据加密的问题。这个工作组当前正在研究覆盖面相当广泛的规范，以便最终提供更加安全的 IP 功能。在 http://www.ietf.cnri.reston.va.us/html.charters/ipsec-charter.html 站点你可以找到有关这个组织的更多信息

续表

JEPI（Joint Electronic Payment Initiative，联合电子支付协议）	它是由 W3C 创建的标准，JEPI 提供了创建电子贸易的方法。使用电子现金或信用卡进行交易，从客户到服务器间的数据传输将使用加密、数字签名、身份认证（密钥交换）来保证安全的交换。这是一个新兴的标准，它的一些条目，比如传输层安全（也叫私有），当前正通过 IETF 继续向前发展
Kerberos 网络认证服务（V5）IETF RFC1510	这是个已经批准的 IETF 标准，它定义了第三方认证的协议。Kerberos 模型部分基于 Needham 和 Schroeder 的可信任第三方认证协议和基于 Denning 与 Sacco 提出的第三方认证协议的修正协议。与许多 Internet 认证协议一样，Kerberos 作为一个可信任的第三方认证服务来工作。它使用了依赖于把共享的公共密钥与私有密钥结合起来的常用加密方法。Kerberos 着重于带可选服务器认证的客户身份认证
Privacy Enhanced Mail part I（PEM1）消息加密和认证过程 IETF RFC1421	这是个已经批准的 IETF 标准，它用来确认你的私有邮件依然保持秘密。实质上，它显示了一个你在加密邮件中采用保护方式的过程，这些过程在解密过程中用户是看不见的。这个标准运用了密钥和其它形式的认证管理。此标准的一些条款基于 CCITT X.400——特别是采用了邮件句柄服务（MHS）和邮件传送系统（MTS）中的某些条款

续表

Privacy Enhanced Mail part II（PEM2）基于证书的密钥管理 IETF RFC1422	这是个用于管理安全密钥的已批准 IETF 标准。它提供了基于公共密钥认证技术的基础结构和管理结构。IETF RFC1422 是 CCITT X.509 规范的增强改进版本，它为使用 PEM 提供了密钥管理基础的过程和协议，这是 CCITT X.509 中所没有的
Privacy Enhanced Mail part III（PEM3）算法、模式和标识 IETF RFC1423	这是个已经批准的 IETF 标准，它定义了加密算法、使用模式和 PEM 特定用法的标识。这个标准覆盖了与加密信息相关的四个主要领域：信息加密算法、信息完整性检测算法、对称密钥管理算法、非对称密钥管理算法（包括对称加密和非对称签名算法）
Privacy Enhanced Mail part IV（PEM4）密钥证书和相关服务 IETF RFC1424	这是个已经批准的 IETF 标准，它定义了验证密钥的方法，它也提供了一个与加密相关的服务列表，Internet 站点可能需要把它提供给最终用户

续表

安全/多用途Internet邮件扩展（S/MIME）	这是由多家公司联合提出的一个标准，这些公司包括Microsoft、Banyan、VeriSign、ConnectSoft、QUALOCOMM、Frontier Technologies、Network Computing Devices、FTP Software、Wollongong、SecurWare和Lotus,它最初由RSA Data Security公司开发，目的是为了使不同产品的开发者能使用兼容的加密技术创建消息传输代理（MTAS）。它实质上意味着，如果有人用Lotus产品发送给你一个信息，你能使用Banyan产品来读取它。S/MIME以流行的Internet MIME标准（RFC1521）为基础
安全/广域网（S/WAN）	在很多人眼里，S/WAN现在只是一点朦胧的微光。S/WAN是RSA Data Security公司提出的一个标准，IETF也有一个委员正在研究这个标准，RSA准备将IETF的IPSec标准也纳入S/WAN中。S/WAN的主要目标是允许各公司最佳地混合和匹配防火墙与TCP/IP栈产品，以创建基于Internet的虚拟专用网（VPNs）。这两种产品当前的方案通常是将用户封锁在单一资源内

续表

SHTTP（安全的超文本传输协议）	这是 Open MarketPlace Server 公司当前使用的加密数据传输技术。在功能上它与安全套接层（SSL）非常相似，而最大的不同是这种方法只应用于 HTTP。现在还没有标准化组织支持这种标准，不过将来会有的。（IETF 最近成立的 Web 事务安全组织，正关注着这方面的技术）
SSL（安全套接层）	SSL 是一个 W3C 标准，它最初由 Netscape 公司提出，目的是为了让客户至服务器端能从协议层传输加密信息。套接口允许在高层协议如 HTTP、NNTP 和 FTP 上进行低层的事务加密。这个标准也制定了客户和服务器的身份认证方法（客户站点的身份认证是可选的）。通过 http://www.netscape.com/info/security-doc.html，你可以了解这个标准的详细信息
WWW 上的统一资源识别（URI）IETF RFC1630	URI 是 IETF 正在开发的标准，目前，资源名和地址是纯粹的文本。URL（统一资源地址）实际是包含了地址信息的 URI 的一种形式，这个地址在 Internet 上映射到一个指定的位置。URI 能提供将 Internet 对象的名字和地址编码的方法。实质上，要访问一个私有的站点，你应该知道它的编码名字，而不是纯粹的文本名

注释 MTS 是个有多种意思的缩略语，在本书中只用到了两种：微软事务服务器和邮件传输系统。除了第 14 章中代表邮件传输系统外，一般

都是指微软事务服务器。

Web 链接 表 14.1 中为你介绍了很多由 IETF 创建的与安全相关的标准。大部分 IETF RFC 文档能在 http://ds.internic.net/rfc/上找到。在 http://www. ietf.cnri.reston.va.us/html.charters/上，你也能看到 IETF 当前工作组的列表，这些工作组协助创建 Internet 上使用的标准。

表 14.1 中的安全标准描绘了 Internet 最终的安全蓝图，将它牢记在头脑中是很重要的，所有这些标准真正覆盖了客户和服务器间的连接。你还能在客户、服务器端增加其它安全策略。这些标准中的大部分还没有覆盖 Internet 的附加产品，比如防火墙等。你的公司能够增加这些附加的安全特性，并最终使得黑客很难攻进你的系统。

Web 链接 如果你还没有足够多的安全信息的话，可以花点时间查看一下 IETF 的情报站点，目前的两个站点是 gopher://ds1.internic.net/00/fyi/fyi8.txt（站点安全手册）和 gopher://ds1.internic.net//00/fyi/fyi25.txt（网络信息检索状态报告）。

谁是 W3C

在我们做进一步学习前，让我们大概了解一下 W3C 组织。W3C 最早出现在 1994 年 12 月，当时，它批准了 SSL（安全套接层）；在 1995 年 2 月，W3C 也支持了 Internet 上的应用级安全。它当前的项目是数字签名协议，W3C 于 1996 年在巴黎发表了该协议。正如你所看到的，W3C 是一个专注于安全需求的标准化组织，它的其它一些职能也是从这个角度发展起来的。

技巧 如果立即使用安全套接层（SSL），将还有许多安全方面的顾虑，因为一些计算机黑客在特殊情况下有可能破解你的密码。美国.政府限制出口加密技术中的密钥长度不能超过 40 位，所以计算机通过尝试 2 的 40 次方的密钥组合，便可以破解这个密钥；美国本土使用的加密程序只能使用 128 位密钥，它意味着一个人要等待计算机做 2 的 128 次方的字母组合才能破译它（这恐怕一辈子也等不到）。那么你将如何判断交易的安全等级呢？Netscape 使这个问题变得非常容易，你只需查看一下屏幕左下角的钥匙：一把断开的钥匙表示没有加密；一把单排齿的钥匙表示 40 位的加密；一把双排齿钥匙表示 128 位的加密。Netscape 和 IE 都提供了文档属性对话框（在“文件”菜单中可以找到它），你可以简单调用这个对话框来查看指定站点提供了什么类型的加密保护方式。

那么我们为什么还需要其它的标准化组织呢？其主要想法是把业界的主要厂商集中起来，创建一个安全的 Internet 网络环境。另外，某些标准化组织——

比如 IETF——主要由自愿者组成，它们的动作过于迟缓，根本不能满足企业的当前需求。W3C 是在为增加新标准并尽快将其实施而产生的组织，W3C 最终提供了实际的标准和规范说明，并提供了采用这些标准的应用示范，以及使用这些标准创建应用程序所需的代码等。

W3C 所研究的另一个项目是 Joint Electronic Payment Initiative(JEPI)，一个将影响 Internet 商务交易的标准。它大胆地假设在英国、法国、德国和美国的客户都能访问同一个站点，并以完全相同的价格去购买同一个商品。但在此之前，它们必须解决一些问题，例如，你是否要询问客户提供一种付款的方式？你需要什么样的付款？你还必须考虑到纸币以外的东西，因为纸币还不能通过电子方式交换。信用卡提供了一种解决方案，但 W3C 也在探究其它方式，如电子支票（electonic check）、借贷卡（debit card）和电子现金（electronic cash）等。这只是其中的一个小主题，更大的主题是要为商品的价格创建一些与纸币无关的方法。例如，你如何告诉一个在德国的人，外套的价格是 80 美元？如果不给他们展示一美元是多少德国马克，他们是否也能知道？对美国人来说，他们若访问一个英国商店，则必须找到兑换英镑的价格。

那么 JEPI 将对安全性做些什么工作呢？它将交易作为一个整体来考虑。你的公司也许不用与一个想买外套的客户打交道，但有可能要与其它一些和贸易有关的问题打交道。大公司也许想提供一个更快、更有效的方法来将资金从世界的一个地方转移到另一个地方。通过 Internet 做电子贸易将最终解决这个问题，这当然不是眼前所能做得到的。如果你的移动雇员缺乏资金时，你该怎么办呢？你可能想到立刻发行一个信用卡，虽然你可以通过 Internet 获得一个日消

费的报告，但现在这样做是不明智的，因为 Internet 还存在太多的安全漏洞。而这些交易标准将帮助公司建立一些新的方法来传送敏感数据。

处理货币的标准方法

如果你的公司在 Internet 上出售产品（不论是有形的还是无形的），那你必须找到一个能在 Web 服务器上收钱的方法，就像在其它地方一样，你能在 Internet 轻松地出售装饰品，甚至还可以卖得更好，但你还得提供一些服务，例如，你要为客户提供一种电子支付帐单的方式，而不是通过邮件送传支票。有谣言说，可能至少有一家电话公司正在作 Internet 监听。小公司可能会发现 Internet 交易太麻烦，但一些中等和大型企业将在某些方面倾心支持它，银行和信用卡公司最终也将加入其中。事实上，通过在线填写的形式，你便能获得一个 Web Conductor VISA 卡（查看站点 http://www.conductor.com 以获得详细的信息）。这个特制的卡允许你在线查看状态和请求服务，在不久的未来，它将允许在线付款，以大大减少填写支票所花的时间。

当前有三种不同的公开方案来实现在 Web 服务器上处理现金，它们是 First Virtual Account、DigiCash 和 Cybercash。每种方案都各有千秋，也各有缺陷，你应该认识到的是，它们中没有一个是至善至美的，如果现在实施这些方案，你可能需要在某些地方做一些妥协。虽然有许多金融公司正在研究一些方法来确保金融贸易的真正安全，但恐怕目前尚无有效的方法。

从实施的角度看，First Virtual Account 方案更容易使用，用户通过电话报名加入一个 First Virtual Account 并接收到一个指定的帐户名，这个帐户名与信

用卡提供的帐户名毫无关系，支持这种方案的商家使用安全证书来接受这个指定的号码。每个交易都使用这个号码，而不使用真正的信用卡号码，这样防止了有人偷去你的信用卡并趁你不知道时使用它。商家通过 First Virtual Account 做交易，它能检验用户的事务，当用户同意交易，商家便接收到付款并开始运送用户所需的产品。你能在 http://www.fv.com 站点上看到有关 First Virtual Account 的更多信息。

DigiCash 取代了信用卡而使用真正的资金做交易，这意味着商家不必支付信用卡的追缴款。另外，DigiCash 还可以像 ATM 借贷卡一样工作，商家获得了真正的资金，而不用等待信用卡公司来按期支付，这需要用户在一个指定的银行有抵押，银行设置一个帐户，除了在线工作外，这个帐户就像是一个支票户头。当用户购买物品时，他或她使用“E-cash”，商家简单地进入 E-cash，便可获得真正的资金。显然，用户必须不时地添满自己帐户内的资金。你可以访问 http://www.digicash.nl/站点来获取更多信息。

Cybercash 是前面两种方案的组合，它提供了借贷和信用两种能力，有点像银行内有透支保护的户头。与其它两种方案不同的是，Cybercash 使用真正的帐户信息，对用户来说，这有点冒险，因为他们的帐户信息能在任何地方以非加密的形式终止。当用户购买东西时，Cybercash 弹出一个窗体，要求输入帐户信息，在信用方式下，用户要提供自己的名字和信用卡号码；在借贷方式下，用户要提供本地银行的名字和其它银行的帐户名。你可以在 http://www.cybercash.com 站点看到与此相关的更多信息。

Web 链接　不止是银行和信用卡公司在讨论在线交易问题，作为一家 Web

服务器公司，Open Market 也在制定基于 SHTTP 的在线交易方案，如果你使用了这个方案，那么它将充当银行和信用卡公司。你可以在 http://www.openmarket.com 站点找到与此相关的更多信息。

第 15 章　建立帮助文件

概述一般原则总是要冒风险的，特别是对一本书。程序员最头疼的任务是书写文档，但是比起写帮助文件来，那可算得上是美差了。当用户使用软件时，帮助文件做得很糟糕，就自然会去读随机文档，尽管后者也不见得好多少。大多数情况下，用户总是抱怨帮助文件只是做做样子，而不是真正对使用软件起到帮助作用，其实程序员已经耗尽心机了。

最近，帮助文件的做法引入了一个新的热点——基于 HTML，结果是更添乱了。Windows 的帮助文件机制尽管不完美，但它提供了一个完善的搜索引擎，用户可以自行查找想要的条目。其不足之处在于某些情况下软件更新太快（比如操作系统），而相应的帮助文件难以及时更改。基于 HTML 的帮助文件的好处在于易于更新。在向电子商务迈进的时代，灵活易变的帮助文件越来越重要了。

回想一下，我们会发现帮助文件与应用程序本身一样重要。如果用户面对庞大的程序不知从何下手，程序员即使挖空心思设计了一个得意之作，也不会得到如潮的好评。这时一切都成了无稽之谈：新增了好多功能，有什么用呢？用户根本找不到！这些足以说明优秀的帮助文件对一个软件来说是何等重要啊！

您该做的最后一步就是书写帮助文件了——想躲也躲不掉的任务！应用程序越大，帮助文件也就越长越复杂。通常 4 个小时的程序开发工作量需要 1 个

小时的建立帮助文件时间。您可以缩短帮助文件的编写时间，但切记不可偷懒。帮助文件中做的成功的部分是那些作为程序用户手册的那部分。我认识的某些人就使用同一个文件既做联机帮助、又做用户手册。

注 基于 HTML 的帮助文件易于更新，使用灵活，有些厂家用它来代替印刷的说明书。

注 帮助文件是应用软件的指南：帮助用户快速掌握软件的使用方法。

Web 链接 如何把帮助文件组合起来呢？本章主要阐述这个问题，这时可能要和其它程序员协作。有些站点专门讨论帮助文件，例如：comp.os.ms-windows.programmer.win32 和 microsoft.public.win32.programmer.tools 新闻组。还有一些站点也有讨论，如 microsoft.public.access.chat 新闻组专门讨论数据库应用程序的帮助文件。如果开发 Internet 应用，那么 microsoft.public.internet.news 新闻组很有帮助（在 Internet 应用中，基于 HTML 的帮助和 Windows 帮助都可采用，但前者更普遍）。甚至可以在一般的编程讨论组如 comp.programming 中加以讨论。

注 编写帮助文件首要的一步是列出大纲，就像编程一样。

以下各节将会帮助您解决许多编写帮助文件中遇到的难题。目前有许多编写帮助文件的技术，所以必须首先确定哪种技术最能满足您的需要。本章的第一节将会谈到两种技术，并给出各自选用的根据。显然，无法定义一种放之四海皆准的标准，但至少有些基本的出发点需要考虑。

像所有编程工作一样，当我们着手建立一个帮助文件时，最先考虑的就是

如何组织的问题。如果您的思路清晰明确，写起帮助来就会得心应手。事实上，您要做的就是把程序流程通俗化、条理化。

接下来就该考虑如何实施了。本章介绍了两种 Windows 帮助文件的实现方法。第一种使用 Help Compiler 加 RTF 文件实现，当帮助文件中有许多唯一特性，而且要快速查找时，这种方法很适用。第二种方法基于 Microsoft 的 Help Workshop，这种方法的好处在于省时易学。

只有将 Windows 帮助文件集成到应用程序中后，帮助文件才能发挥作用，本章的下一节将讨论这个问题。我们考查几种不同的方法为第 2 章中建立的应用程序编写帮助文件，您会看到如果要具有上下文相关帮助功能，帮助文件将变得非常庞大，如同小驹配华鞍，喧宾夺主！

Web 链接 也有相当一部分人放弃采用 Windows 帮助文件模式，转而使用 HTML 文件，就连微软目前也在它的部分产品中使用了这种方法。HTML 文件的主要缺点是搜索功能不强，格式不够规范；其好处则在于内容更新十分方便。如果想进一步了解这方面的情况，可查阅 HyperNews Forums Web Mastery Resource List，其网址为：http://union.ncsa.uiuc.edu/HyperNews/get/www/html/guides.html 这个站点中包含了许多内容，指导您将 Windows 格式的帮助文件转化成 HTML 格式。还可以参看 WWWWAIS.C 站点，网址为 http://www.eit.com/software/wwwwais/wwwwais.html，WAIS 是个让你在 Web 站点中加入搜索功能的程序。

基于 HTML 的帮助文件也需要最后的实施阶段，然而其实现方法更像是设

计一个 Web 主页的过程。在阅读本章之前，请先读第 8 章，从而对 HTML 概念有一个基本的了解，本章只讨论建立基于 HTML 的帮助文件时，有哪些需要特别考虑的地方。

本章的最后一节将演示如何把基于 HTML 的帮助文件集成到第 2 章所建立的应用程序中。尽管不是很难，但还是比集成 Windows 的帮助文件要复杂一些。另外，你会发现 HTML 在生成真正的上下文相关帮助方面存在一些限制，但可以采取一些迂回措施达到这一目的。最后本节还将讨论为什么有许多的厂家喜欢使用 HTML 做帮助文件——就是为了便于更新！

15.1 确定要创建帮助文件的类型

微软试图使人们相信基于 HTML 的帮助文件是有史以来最好的帮助形式，也是您开发新的应用程序时应该唯一考虑的选择。然而帮助文件形式的选择是由一系列因素决定的，而不只是为了赶时髦。

注释 寻求基于 HTML 的帮助文件已经成为商业软件的事实标准，原因在于它易于建立、减少了编辑使用手册的开销。然而少花钱并不等于这种文件就好使了。

事实上，许多微软的用户正在抱怨基于 HTML 的帮助文件对于微软的某些产品来说很不实用。当然也有一些情况则很适合用这种形式的帮助文件。那么，我们究竟怎样取舍呢？我们最好还是先对二者加以比较，以下就是选择 HTML

的原因：

升级能力 建立 Windows 帮助文件并嵌入应用程序后，帮助内容就相对固定了，因为一旦装入了用户的机器就不好再更改了。基于 HTML 的帮助文件驻留在某一 Web 站点上（包括 Internet 和 Intranet），所以一经更新，用户总是看到最新的内容。

减低了学习难度 随着 Internet 和 Intranet 的日益普及，企业中只需几个人掌握建立和更新 Web 主页的技术，专门负责维护就可以了；而 Windows 帮助文件则需要每个程序员都理解其编制规则，显然浪费了劳动力。

减少了与语言\特殊需求相关的问题 有许多种方法使得 HTML 帮助文件支持多种语言、面向特殊应用，而不必考虑 Windows 帮助文件的影响。可以先询问用户想用哪种语言，按照回答提示用户转入相应的站点去查阅。对文种支持的自动化使得不必在每个应用程序中再考虑文种的重定向问题。应用类似的技术也简化了其它一些需求，比如用户需要支持大字体等。

增强了定制化的机会 HTML 的开放机制使得定制帮助文件非常容易。Windows 帮助文件中，所有的图形都存在于编译好的文件之中，而基于 HTML 的帮助文件则不然，可以随时提取其中的图形，这就意味着可以按照用户的要求进行度身定制，如更改公司徽标啦、反映企业形象啦等等，一切信手拈来。

然而这并不是说 HTML 帮助就无所不能了！事实上，它也带来了一些新的问题。以下就是使用 HTML 帮助文件时所不具备的特性：

安全性　使用 Windows 帮助文件时，可以隐藏一些信息只让管理员可以查阅。而 HTML 帮助文件都是文本形式，所有人都可以知道是否有管理员用户信息存在，并试图访问这些信息。

大小　Windows 帮助文件可以进行压缩，通过减少磁盘碎片来节省硬盘空间；而 HTML 帮助文件则无法压缩，磁盘空间将是个问题。

远程访问　Windows 帮助文件就在本地硬盘上，随时可用。而 HTML 帮助文件的可取之处也正是它的致命之处：如果用户无法访问 LAN 或 Internet 时怎么办呢？这时本机上拷贝的帮助文件即使已经过时，也变得异常宝贵了。

没有注释＼降低了书签支持　用户具有向帮助文件中增加注释的功能非常重要，然而这也是 HTML 帮助文件所不具备的。理论上可以允许用户修改原始的 HTML 文件，但比起 Windows 帮助文件来要复杂多了。相似地，即使只是在 HTML 帮助文件中加入一个“收藏夹”特性（类似于 IE 中的收藏夹），用户也只能保留该主页的地址。而 Windows 帮助文件可以随心所欲地在帮助文件中做标志，充分发挥书签的作用。

减低了检索能力　前面我们已经谈到了这个问题。Windows 帮助文件可以很方便地查找指定的单词。关键字的应用使得用户可以建立自己的查找标准。而 HTML 帮助文件不说是绝对不行，也是很难做到这种查找的，肯定不能查找单个字词或关键字。

说到这儿，您可能已经在权衡两种帮助文件的利弊了。的确它们互不相同，各有所长。我们选择的依据何在呢？您可能还出于一些我没有讲到的考虑来作

出自己的决定。

15.2 组织帮助文件

设计帮助文件有许多方法，如何组织是首要任务。组织的好坏是使用难易的决定因素。例如：面向任务的帮助文件可以指导用户迅速地完成任务，当然其前提是程序设计者很清楚自己的应用是干什么的。而面向菜单的帮助文件可以指导用户快速查找菜单条目。

帮助文件的类型也会影响其组织形式。例如：Windows 帮助文件提供了强大的查找能力，这时您就会更多地用到面向任务的组织形式，以便用户能够随时查找命令信息。此外，使用 HTML 很难建立教程性质的帮助文件，采用菜单流控制的组织形式则相对比较容易，因为新增菜单项比在 Windows 帮助文件中实现还容易。

建立大纲

着手建立帮助文件的最重要一步就是编写大纲。大纲应反映出用户的需求，这有助于从一开始就组织好结构，从而把写作重点放在具体内容上。下面是我个人的一些经验，供大家参考：

菜单流 菜单流技术以应用程序的菜单系统为中心，从它开始并作为主线。先将所有菜单项列成层次清晰的结构化条目，再列出所有对话框；每个对话框下要包含其中用到的所有控件；最后，列出主窗口及其组件。

使用这种组织方法有利于用户快速定位到应用程序的某一具体条目，缺点是用户往往只看到该条目的含义，却还是无法完成想要实现的任务。因此，这种方法适用于用户已经有一些相关经验的情况——只需要告诉他们本应用程序的用途，不用再从基础一一教起了。这种帮助文件很适合于实用程序，因为这一类型程序的用户购买该软件时目的明确，他们知道买来做什么。也适用于配置程序模块，这时的问题直接了当，但用户还是需要一些帮助的。

任务　我接触到的大多数用户并不关心当今程序中新出现的“gee whiz”特性。用户只是希望在最短时间内能最快地完成工作，特别对那些整日在公司里忙碌于琐事的职员，他们没有时间去学习如何做某件事，你只能告诉他们。这时，这种帮助文件就很有用了。先列出用户可以实现的所有任务，再一一加以解释。建议您从任务本身的解释开始，用户可以完成什么功能呢？再告诉用户采取什么步骤完成它。这些用户如果对计算机一无所知，而且毫无兴趣倒反而好了！这种技术适用于数据录入和任务固定的应用程序的情况，而对于字处理和其它综合性的应用程序就不能胜任了。

通用菜单／任务　有时所开发的应用程序执行许多任务。例如：一个字处理程序不只是一个目的，而有多种用途。如果此时您宏篇巨箸地逐一解释每项功能怎么使用，对您和用户都是一件残酷的事。我常用的做法是：对各种任务先进行分类，再对每类给出常规指导，而不是对每项任务加以解释。对菜单系统给出总缆介绍也值得推介，这样用户就能很快

了解应用程序的用途而不必陷于揣摩编者的构思之中了。由此引发了“气球帮助”的大量应用，这种气球是一种标志，鼠标在某一控件上停留几秒钟时它就会出现在鼠标指针旁边。

参考　编译程序和自带宏语言的应用程序经常使用参考手册类的帮助文件。这类文件是从控制的角度而不是具体结构上分析程序的。对一个字处理程序来说，按字母顺序列出宏命令清单就足够了，因为无法预测用户会怎么使用这些宏。通常还会具体描述这个命令，详述每个参数的含义，并给出一些用法的例子。显然也可以加入其它信息，如：提示不同版本的特殊性。用户常感到有用的信息有：“如果想做X，请使用此命令；否则请使用命令Y！”，这非常有效。

教程　这是一类特殊的帮助文件。只用于教给初学者，实质上是要告诉他们如何使用您的程序。多数程序员觉得：当用户毫无经验时，这种帮助文件非常有用。用户只需稍懂计算机使用常识就可以了。我发现适用于使用这种帮助文件的情况之一是数据录入程序。一般我先对某项具体任务给出一个简要的解释，再给出一些使用问答。如果能对实际应用进行抽象，给出交互式的帮助会话也非常可取。可以用帮助文件的宏命令来实现，它指导用户输入正确的操作。这一类型帮助的不足之处在于：大量的主题难于组织在一起，而且用户经过长时间的操作可能还搞不清楚到底要做什么。费时费力地做这种集成还不如努力说服公司雇一、二个培训人员呢！

功能区　一些应用程序提供这种帮助是因为它们特殊的用法。比如CAD

及其它的绘图程序，专门开辟有这样的功能区。CAD 程序中有绘图命令组、颜色模板控制命令组及图形大小形状控制命令组。将许多命令分组，有助于用户快速查找。原因很简单：用户绘图时只关心绘图命令，等到要进行润色加工时就只关心颜色啊、效果啊等相关命令。

Web 链接 如果您还是觉得编写帮助文件太麻烦了，可以使用现成的生成帮助软件。SheepNet 的站点 http://www.sheepnet.demon.co.uk/helpfile.htm 就提供了建立高质量帮助文件的方法。当然您还是需要花时间学习怎么使用，而且要想最终给出一个很好的帮助文件还是需要下一番功夫的。

建立脚本

注 帮助文件一般保证在 800 x 600 的显示方式下不超范围。

大纲建立好以后，就要补充血肉了！这时，也有一些规则应该遵循：首先，尽量使一节帮助的内容只占一屏（这种“一屏规则”对 HTML 帮助文件尤其重要，原因在于它不支持书签和注释，用户在第一次找到信息后翻到其它页再想快速查找先前看过的页面就不那么容易了）。用户可不愿意为了查找一个信息来回翻页。大多数情况下，可以将大块的帮助文件分成多个子标题，以便于查阅。在目前的硬件条件下，一屏的大小为 800 x 600（不久可能是 1024 x 768）。虽然有了更大尺寸的显示器，也还会有很多人用小的显示器。

技巧 大量新型笔记本电脑使用 1024 x 768 的显示方式，而老一些的用 800

x 600，甚至 640 x 480。在编写帮助文件时，要牢记这些条件。

“一屏规则”也有例外，例如：不想把一个过程分成几个子标题。谁碰到这样的帮助都会很恼火：像“建立邮件合并主文档——参见标题 3A 来实现”，您说烦不烦？类似的指南中的问题也常见于一些日用品的说明书中，比如音响的说明书。人们可能都听过这句话：“这东西读起来怎么跟音响说明似的，晦涩难懂”。

当不便于分开的时候可能也会打破“一屏规则”。例如：为了解释的直观性可能要加进图形，这时一屏就放不下了。多数情况下复杂的过程要调用多个屏幕，不要因为过程的复杂性而不使用子标题，只要有可能就尽量简化。所要做的是在过程的完整性和分割性之间取得平衡。要保证没有子标题的屏幕也同样容易阅读。我们常常把一些信息放在子标题下，这样用户看到的是一屏完整的信息，想阅读具体内容时只需再点击个别标题就行了。

以上我所说的可以总结为一句话：建立脚本。可能要做的工作像是写本书，但又不能用书的格式。不需要像书那样一个主题紧接着另一个（如果真把帮助文件做成书的形式不知要激怒多少人！因为有些主题只是介绍并没有实际意义），而要求每个主题都实有所指。当然，主题之间加入连贯性将有助于用户使用。

打脚本只进行了全部工作的一半，接下去就要将脚本变成 Windows 帮助引擎或 HTML 可识别的格式，从而用户才能使用。这时要将大纲分屏，并在需要的地方加进超文本链接。也有一些帮助文件编译器需要类似 make 文件的东西，用来告诉编译器在帮助文件中都包含哪些文件，并且可以用按钮进行选择。

以下各小节将完整介绍从脚本生成帮助文件的过程。之后，再介绍如何为用 Visual C++开发的应用程序制作具有上下文相关功能的帮助文件。那是建立帮助文件的最后一步。在把帮助文件放入应用程序之前应该确保所有的准备工作都已经到位。

高级技巧

词汇表的重要性

我们这些整天坐在机器前面编程的人随时会学到许多新的术语，这些术语就像是每天的调味品，而不会成为交流的障碍。常常自己不知不觉中就使用了没有专门去学习的术语。

而对多数用户来说计算机只是个半熟脸，不过是工作中用到的工具而已。（我的一个好朋友甚至把她的计算机叫‘死对头’，多好的形容词！）对这些人来说，新的术语就是障碍了。充满了晦涩难懂术语的帮助文件比没有帮助还糟糕，因为用户费劲读了半天啥也没弄明白。

我们必须面对这一事实，写帮助时不可能不用到术语，我们这个行业几乎每天都在产生新的词汇，只能尽量避免使用太专业的行话，却不可避免地要用到标准的计算机词汇。

一般在提交正式稿以前要打草稿，换句话说，先写个提纲，再逐步完

善。我想要说的是一个简单的书本的格式。这时，可以请一、二个从没有看过的人读一遍，记下他们不明白的词汇，这些人最好是非专业的，这样得到的信息才更真实。做完以后，把这些词汇汇编成表，就是一份很好的词汇表了，比你自己总结的要实用得多。

最后一步，将生词的每次出现都和词汇表进行热链接。这样，用户不知道生词的含义时只需单击链接处就可以了，Windows Help 或 HTML 浏览器就会自动转入词汇表的相应入口处。

15.3 使用 Microsoft Help Compiler

本章我不想介绍 Microsoft Help Compiler 的全部使用过程，因为建立帮助文件只需用到部分命令就可以了。而且，这个软件本身的帮助文件做得非常优秀，是程序设计人员学习的典范。我们要给出一个简单的例子，教您如何用标准字处理软件和 Help Compiler 组织帮助文件。目前为止，市面上还没有更好的工具，我们只能这么做。这也是建立帮助文件最灵活的一种方法，因为设计者对内容有最大的控制权。

注释 本节只讨论 Windows 帮助文件，如果想了解 HTML 帮助文件的技术，请参见本章以后的介绍。

Visual C++在建立帮助文件前还必须多做一步，所幸的是，这一步会为以后减少工作量，特别是在开始做 RTF 或 make file 之前。用 VC++的 BIN 目录下的

MakeHM（制作帮助映象）程序建立一个帮助映象文件。（对于 Visual Studio 的用户，该程序放在 Program Files\Microsoft Visual Studio\Common\Tools 目录下），命令格式如下：

```
MAKEHM   ID_, HID_, 0X10000 RESOURCE.H  MY.HM
```

这样就会得到如程序列表 15.1 所列出的应用程序的帮助标志符，用于完成在 VC++和帮助文件之间的连接。现在，我们只需知道在帮助文件中将包含这些标志符，而且在帮助文件的脚本中也会用到。

程序列表 15.1

```
HID_CANCEL_EDIT_CNTR            0X18000
HID_CANCEL_EDIT_SRVR        0X18001
HID_UNDERLINE               0X18006
HID_STRIKETHROUGH           0X18007
HID_BOLD                    0X18008
HID_ITALIC                  0X18009
HID_FONT_DIALOG             0X1800A
HID_VIEW_FORMATTOOLBAR      0X1800B
HID_FORMAT_FONT2            0X18010
HID_HELP_CONTENTS           0X18012
HID_HELP_WHATSTHIS          0X18013
```

注释 对编辑软件没有要求，就连设计程序环境中的编辑器也可以使用；唯一的要求是脚本文件必须是 RTF 格式的。所幸的是，RTF 实际上就是带有一些特殊格式命令的 ASCII 码文件。使用编辑器时要注意是否支持 CRichEdit MFC 类标准，因为有些老版本不支持某些 RTF 命令。尽管 Visual C++ 5.0 及其以后版本提供的 CRichEdit 还不像标准的编辑器那样专业，但对建帮助文件来说还是可以很好地工作的。

Microsoft Help Compiler（Microsoft 帮助编译器）至少需要两个文件：帮助脚本文件和 make 文件。我通常使用 Microsoft Word 来编写脚本，因为它支持所有的 RTF，而且我用得很熟。首先要将帮助文件分节，每节之间插入一个分页符，在 RTF 文件中，使用\page 语句进行分页（我在讲解过程中会介绍一些 RTF 语句，以供大家参考）。

Web 链接 如果微软的软件还不能满足您的需要，可查阅以下站点：http://win95.daci.net /webwhelp.htm，该站点提供了大量的 Windows95 的 WinHelp 工具和实用程序，例如：Help Maker Plus 程序，专为替代 Microsoft Help Compiler 而设计。另一个很好的站点是 Winhelp WWW Index，地址为：http://www.hyperact.com/winhelp/FTP_and_file_Archives/index.html，虽然该站点不包含直接的信息，却给出了大量的有用链接地址，如：指向 RTF 规范说明的站点，指向 SHED 规范说明的站点等等，还有指向其它帮助文件编译器和设计程序的站点。

接下来就是为每个标题加入一个或多个脚注了，许多超文本链接函数要用到脚注。例如：索引中的查找单词就是由它而来；还用于在词汇表和帮助文件之间建立链接。表 15.1 给出了一些脚注，这些都是常用的。还应该查阅一下 Microsoft Help Compiler 的文档，常会有新的补充。

那么脚注中有哪些内容呢？这要依脚注类型而定。例如：使用 # 脚注，表示要加进超链接。使用前要仔细阅读使用规则，一定要注意脚注名的唯一性。名字最好是描述性的，否则自己也记不住。与变量一样，超链接的名字中不能含有空格。在 RTF 文件中脚注表示为：<footnote type>{\footnote<text>}。

然后要对这些含有标志符（#）的脚注进行编译。有了这张清单，就可以顺利地组织帮助文件建立超链接了。我们先考查一下标准的帮助文件：绿色的文字表示超链接，实际它们是在双下划线或可点击的标识下的文字（RTF 中分别为 /uldb 和 /strike）。所以建立超链接的第一步就是进行这项工作。双下划线后是超链接的标题标识符，用隐含文本书写（RTF 中用 /v 语句实现），这与 # 脚注中的标识符相同。

这时还有一些事情要决定，比如：是否在帮助文件中加入图形。在适当的位置加入图形会大受欢迎，还可以加进声音和多媒体效果，但我觉得最好是在对这些技术应用娴熟之后再这样做，起初还是不用为好。还必须决定在 make 文件中放入哪些内容。以下的章节详细叙述了完成帮助文件的实施步骤，我还会解释各种可用的选项，以及对不同类型的帮助文件我是怎么实现的。

表 15.1　Microsoft Help Compiler 中的标准脚注样式

脚注类型	用途
*	最后机器上可能会留有许多 RTF 文件，但其中所包含的主题不是每个帮助文件都能用到的。举例来说，我为一个用于在线问候的通讯程序做了一个帮助文件，多数用户感觉不错，当他们在线看到陌生的姓名缩写时，借助帮助能很快加以识别。这是用含有多个主题的一个 RTF 文件实现的。这时又要为其它实用程序编写帮助文件，前面文件中除个别主题外，其它主题都适用，该怎么办呢？就用这个脚注标识符，它定义了一个建立标识，与后面将介绍的帮助工程文件（HPJ）配合使用。Microsoft Help Compiler 先检查帮助中要包含的主题，再检查 RTF 文件中的建立标识符。这个标识符必须做为主题的第一个脚注。建立标识是不区分大小写的，但我总是用大写，这样即使 Microsoft Help Compiler 以后改动了也不会受到影响。典型格式如下：*{/footnote BUILD_TAG}
#	用于标识主题的脚注。可以将它看成 GOTO 语句中用到的标号。当用随后介绍的技术跳转到这种主题标识符时，Windows Help 的控制也就转到这儿了。它属于帮助文件中超链接的一部分。超链接可满足多种需要，包括菜单和与词汇表建立连接。它也是不区分大小写的，我还是用大写，原因同上。典型格式为：#{\footnote SOME_LINK}
$	用于建立主题标题的脚注。帮助主题的标题出现在窗口的正文上方灰色区域内。也会在“主题查找”和“历史记录”对话框中出现。可以是所有文本。比如：${\footnote 这是一个标题}

续表

+	有时需要在一系列帮助主题间实现直接转移以方便用户使用，例如：一个长过程即使勉强放在一个窗口也不便于使用，改进的办法是将这个过程用几个可相互转移的窗口来表示。‘+’脚注就是用于这个目的的，可以把它称为浏览序列标识符。它激活了帮助窗口中的两个浏览按钮：<<和>>。Windows 允许使用任何标识符做为浏览顺序，它自动地对标识符以字母序排序；总是用页序号排序。例如：+{/footnote Page:1} 表示该序列中的第一页。要注意对每一个标题只能用一个序号 必须在 HPJ 文件中加入 BrowseButton 宏命令来使浏览按钮起作用。Windows Help 在帮助文件初始化过程中查找这个宏命令。本章中“建立 make 文件”一节将讲述加入的方法 这个标识符是用来方便使用的，用户用它可以将长文分段而不会引起岐义。我发现在显示一个多页图形时也很有用，如层次结构图。例如：某帮助文件包含有 Novell 论坛的结构图，长度超过 1 页，用这个标识符用户就很方便地跨页阅读了。它也适用于参考手册型的帮助文件——它能够从一个命令转到另一个命令。它的应用范围几乎没有限制

续表

K	帮助文件的查找能力依赖于关键字脚注。在帮助文件中为每个主题及其子题目定义一个或多个描述字。我总是犯选过多关键字的毛病。如果用 Windows 95/98，关键字会出现在‘查找’窗口的‘索引’页。关键字可以含有各种字符，其中可以包括空格。Windows 还可以识别关键字的大小写，这样您就可以尽量描述清楚主题的含义，用户也就容易理解了。每个主题中可以有多个关键字，关键字之间用‘；’隔开。 在 DOS 的帮助编译器 HC31 中有一个缺陷，如果使用这个脚注会带来一些麻烦。在 RTF 文件中，这个脚注和其后的文字之间必须插入一个空格，否则它就显示不出来了。而且，如果关键字是以 K 打头的，更得多插一个空格或分号。 使用该脚注的一个实例为：K{/footnote Control; Exit Pushbutton; Leaving the Program}。这时用户能够找到同样的帮助主题：Control, Exit Pushbutton, Leaving the Program。在建立 RTF 文件时维护关键字的有序性有助于使复杂的帮助文件保持一致。一定要确保主题每次出现用的都是同一个关键字。例如：如果某处用到了“Control”，另一处也必须用同样的形式，千万别用复数或其它形式。如果帮助文件前后一致，用户很快就会用顺手，您也就会少很多麻烦，否则就等着到处答疑吧！

续表

@	如果一个程序没有注释会怎么样？当要增加新功能时您将无法指出哪些是原来就具有的。您看看本书中的例子，我总是给出许多注释，因为多年的编程实践使我充分认识到了注释的重要作用。随着帮助文件的复杂化，您很可能会忘记为什么加进了宏命令或其它的做法。标识注释的脚注解决了这个问题。它的用法跟向帮助文件加注释一样，区别在于大部分情况下这些注释不可见，除非以脚注视图打开它（当然前提是：文件是用标准编辑器建立的）。该类型脚注的典型格式为：@{/footnote This is a comment.}。不用说，帮助编译器根本不理会这种脚注，所以您什么都可以往里放

给帮助文件添加特殊效果

可以对帮助文件做许多润色工作。我最常添加的效果就是图形。例如：可从实际程序中抓取屏幕样图，用 Hotspot 编辑器(SHED.EXE)在图上定义热点，该编辑器支持 BMP、DIB、WMF、SHG 等多种图形文件格式。一般含热点的文件都使用 SHG 格式。

技巧 对教程类的帮助文件我最喜欢加入的图形是“回答”按钮。它们看上去就像标准的 Windows 按钮一样。用户看到提问作出回答后，按下这个按钮，就会得知操作得对不对。这个方法也可以应用于其它控件，Windows 标准帮助中没有提供这么强的功能，它提供了：{BUTTON [LABEL],Macro1 [:Macro2:...:MacroN]}的命令用以建立标准按钮。

Label 内容是按钮上显示的文字，宏命令则是实际执行的动作。

下面我们花点时间讲解一下 Hotspot Editor 编辑器。图 15.1 是它的典型界面，这里我打开了一个后面也要用到的例子程序窗口，该程序用于测试帮助文件。目前我们只关心主窗口的格式。我打算为帮助文件的每个控件设置热点（hotspot）以方便用户查找，不能用 Hotspot Editor 建立一个新图，真想那样做只能用其它图形软件；Hotspot 只是按照用户设想的动作在相应位置作出标识。当帮助文件中的光标由指针形状变为手指形状时，你就会看到 Hotspot Editor 的效果。

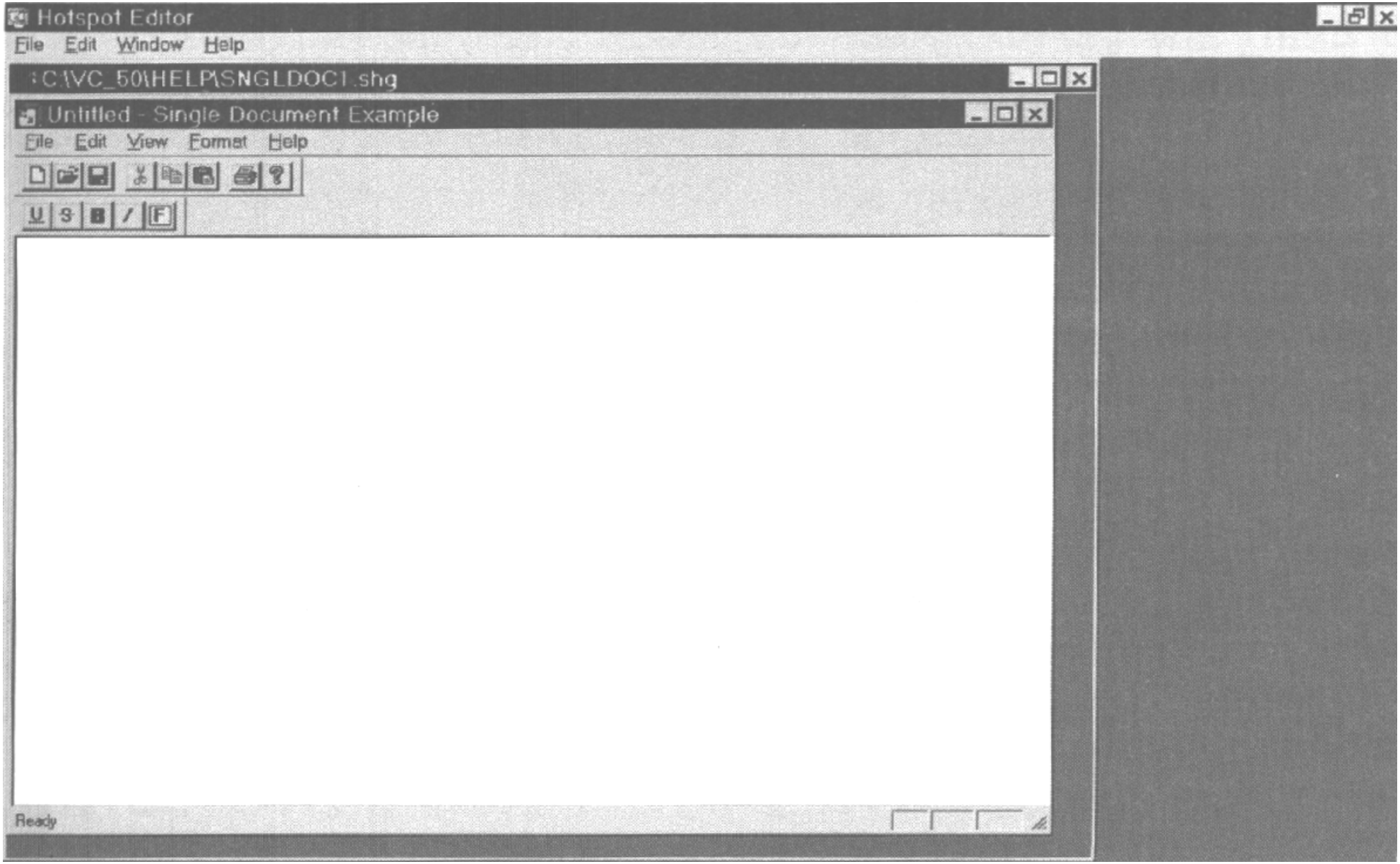
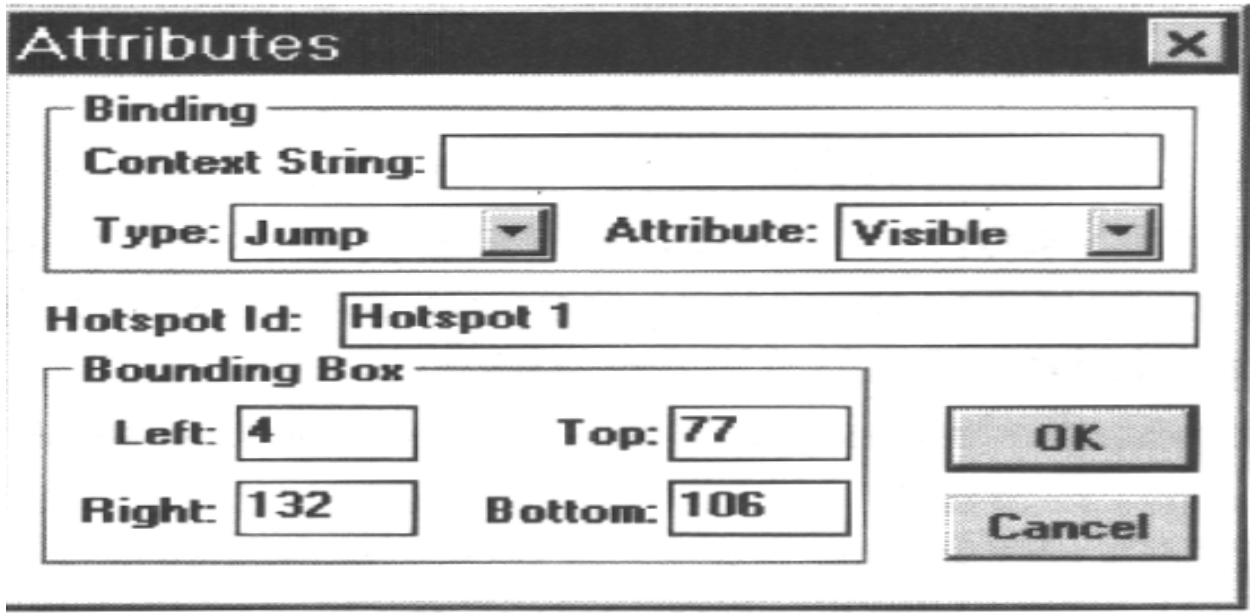


图 15.1　Hotspot Editor 不建立一个新图，只是在现有文件中建立热点

注　属性对话框用于定义为图形所建热点的类型。

建立热点就像拖动鼠标一样简单：全部的工作就是建立一个如图 15.2 所示的矩形框，它指示的区域就是热点。

热点建完还需对它进行定义。双击热点，就会看到如下图所示的属性对话框：



其中只有三个需要指定的参数：内容字符串（Context String），类型（Type）和属性（Attribute）。内容字符串的作用与前面介绍的双下划线标识的作用相同，是超链接的第二种形式，用户点击热点时就会转到相应的指定链接处。热点的边框可设为有或无，我一般都设为无边框。共有三种热点类型，下面分别予以叙述：

技巧 使用 Edit|Preferences 命令可以定义一组标准的热点属性。该命令将打开一个对话框，在该对话框中定义缺省的内容字符串、类型以及属性。使用这个对话框还可以定义缺省的热点标识（ID）。

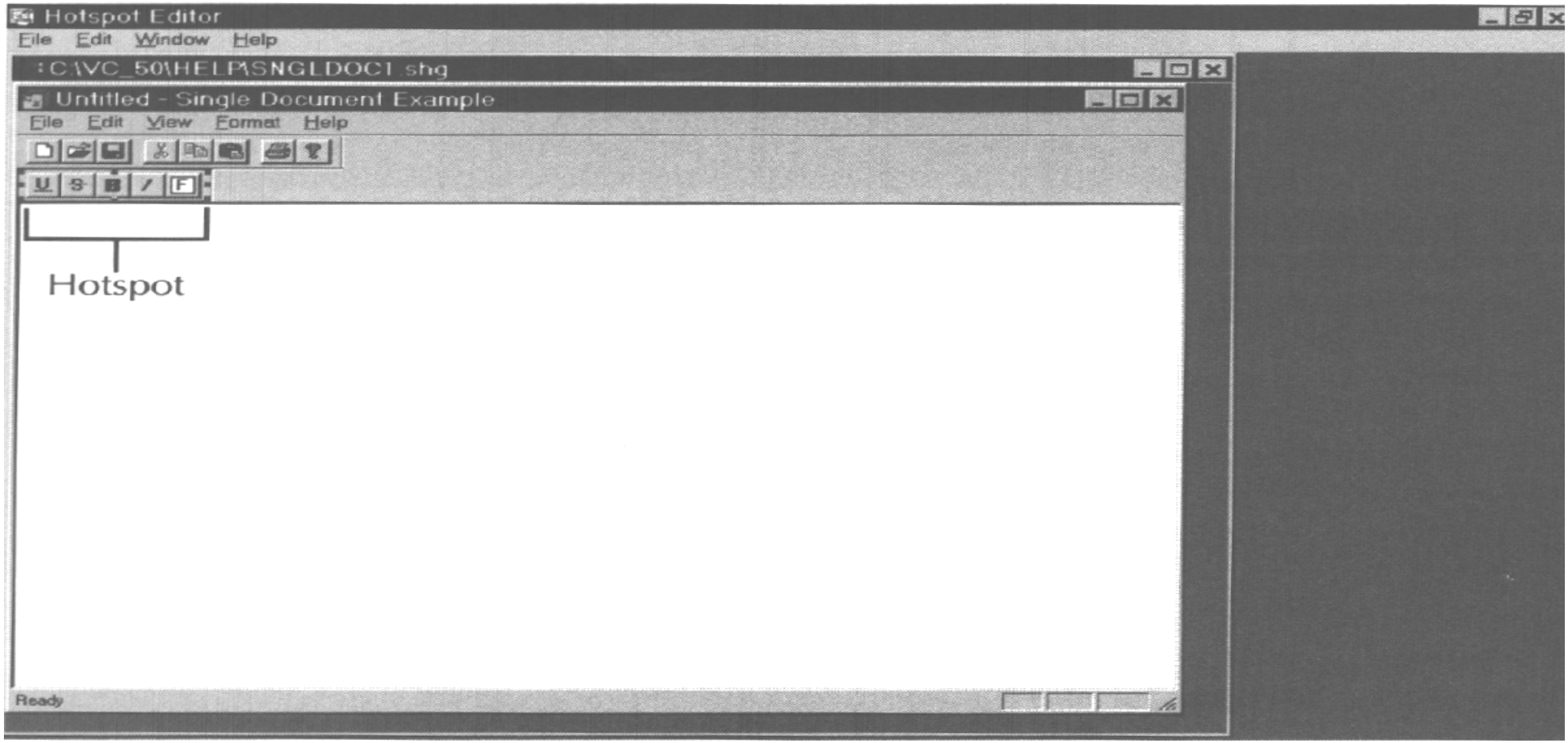


图 15.2 创建热点容易到只要拖曳下鼠标——方框指明了热点的出现位置

跳转型（Jump） 此时，无论内容字符串指示的帮助主题是什么，当前窗口的内容都会被替换，用户实际上从帮助文件的一个区域转到了另一个区域。适用于帮助用户查找同一主题下的多项内容，可以用它建立一个“参见”按钮；也可用于实现在多页结构图中的跳转链接。这种跳转是直接完成的，不需要再按“浏览”按钮。我还用它实现图形控件间的转移，以模拟实际的应用。

弹出型（Pop-up） 这类跳转最常用于控件描述或其它插图性的热点说明。与它相关的帮助主题显示在弹出窗口中，好处是用户无需离开原窗口，就可以方便地查看控件说明。我也将它用于教程型的帮助文件中的问答部分，按“回答”按钮，对问题的回答就会显示在弹出窗口中。

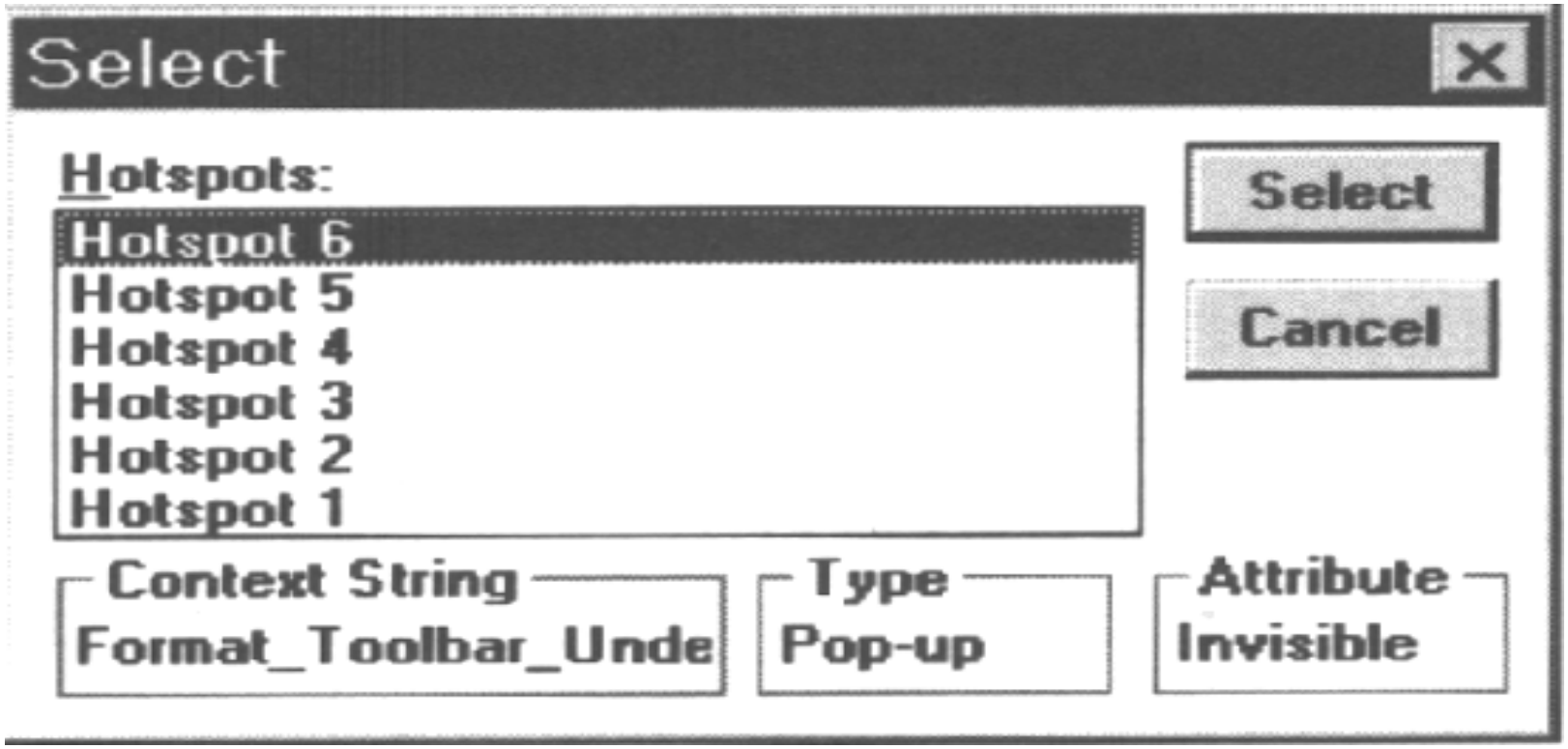
宏命令型（Macro） 可能有时不想让某个图形显示帮助文件中的内容，宏命令型的热点可以执行预定义的宏，后面会给出用宏建立 make 文件的例子，从中体会到 Windows 帮助中宏的强大功能，甚至可以用它改变 Windows 帮助的工作方式。

技巧 Microsoft Help Compiler 提供了许多预定义的宏。例如：SetPopupColor 宏命令改变弹出窗口的颜色。给用户设计一些按钮和菜单，实际是执行这些宏命令，会极大地提高帮助界面的友好性。用位图指示宏的方法是建立彩色方框位图，当用户点击该图时，帮助窗口的颜色会发生变化以匹配其内容。

属性对话框中还有一些有必要了解但无需更改的域。一个就是 Hotspot ID 域，实际是该热点的标识符，由于本书的读者多是程序设计人员，我后面会介

绍一下它的用法。另外四个域定义了热点的边框，用于调整边框的位置和大小，我觉得用它们来调整更方便更迅捷，用户只关心热点是否存在，而不会太在意位置是否偏离了一两个像素点。

有时你可能要重定义某个热点，对于那些可见热点当然好查找，但是我们说过，多数热点都定义为不可见的，那么怎么查找呢？用“编辑 | 查找”命令。它的形式如下图所示：



注意这里显示出了当前图中的所有热点，是按热点标识符顺序列出的，而不是按内容字符串排序的。选中一个热点后相应的内容字符串就会显示，您就

可以验证是否是要找的热点，同时显示的还有跳转类型和属性信息，虽然微软没有加进范围信息，多数情况现有信息足够用了。

全部热点定义完后，可以用“文件 | 保存”命令以当前文件名保存该图形文件，我一般都用“文件 | 另存为...”命令将它存成 SHG 文件，虽然 Hotspot Editor 没有坚持一定要这样做，但我建议最好还是做一下，否则您可能会发现某些更改过的信息用其它绘图软件读不出来。因为用 Hotspot Editor 不能修改图形本身，应该保持一份原文件的副本以便日后修改。

现在我们得到了图样（或其它多媒体元素），怎么加到帮助文件中呢？微软定义了一系列命令用以完成向帮助文件中加入图形或其它元素的功能。这些命令在一定程度上控制着这些控件的放置，但我觉得它提供的位置机制太原始了。表 15.2 是全部命令的清单。

注 一定要用“文件 | 另存为...”命令保存编辑过的图形文件，否则原始图形文件会被覆盖，再想用就没有了。

技巧 可以在表 15.2 列出的三个图形命令前加上 T（透明），它将图形背景的颜色自动变为帮助窗口的底色，从而实现“透明”化。例如：{BMRT FIGURE.BMP}命令在屏幕右上角显示名为“figure”的位图，并将底色变为透明的。该特性仅限于 16 色图形。

表 15.2 多媒体帮助命令

命令	描述
{BMR <Filename>}	在窗口右侧显示图像。必须给出带扩展名的完整文件名。Windows 帮助系统识别位图文件(BMP、DIB、WMF)、多热点位图文件（SHG）和多解析度位图文件（MRB）。但不幸的是，不能使用 PCX 文件
{BMC <Filename>}	在窗口中央显示图像
{BML <Filename>}	在窗口左侧显示图像
{MCI_LEFT [<Options>,] <Filename>}	在窗口左侧显示多媒体控件（MCI 文件）。在 Microsoft Help Workshop 帮助文件中有一个错误，它说该选项只支持 AVI 文件，其实最新版的 Windows 帮助系统支持所有的 MCI 格式，包括 WAV、MID 和 AVI。最好只用这三种格式文件，除非您知道所用格式是现用系统支持的。可以给出一或多个选项，包括：EXTERNAL、NOPLAYER、NOMENU、REPEAT 和 PLAY。 EXTERNAL 选项保证该文件独立于帮助文件，从而减少了加载帮助时占用的空间；不便之处在于多媒体文件必须作为帮助中的独立项目。Windows 帮助在显示文件时总是显示相关的媒体播放器，可以用 NOPLAYER 选项屏蔽它，实现自动播放和重复播放的功能。NOMENU 选项显示一个不带菜单按钮的播放条，它保持了播放条的基本功能但不能重放。REPEAT 选项则实现第一次播放完成后立即重播功能。PLAY 选项进行自动播放——非常适用于闪烁窗口

续表

{MCI_RIGHT [<Options>,]<Filename>}	在窗口右侧显示多媒体控件（MCI文件）

技巧 为支持多种显示方式，可以为同一命令指定多个位图文件。Windows帮助系统自动选用最适合的位图文件。例如：命令{BMR CAT016.BMP;CAT256.BMP; CAT024.BMP}分别提供了16位、256位和24位三种彩色位图供不同的机器使用。更进一步，还可以用MRB文件来提高帮助文件的灵活性，其好处是：您能够非常灵活地显示详细信息，当然前提是用户的机器支持WMF；坏处是：帮助文件骤然增长，大吃内存。

注释 进一步扩展帮助文件的帮助语句是：{EWx<DLLName>,<WindowName>,<Data>}，可以用它访问外部的DLL，其中EWx的x指示DLL中左(l)、右(r)和中央(c)的位置。WindowName包含当前帮助文件窗口的名字，前面我们说过是用$脚注标出的。Data用于向DLL中传递待处理的数据。

注 要慎用图像和声音文件，以防帮助文件过长。

帮助文件中肯定可以有响铃、鸣叫以及其它应用程序能具有的各种多媒体效果。但是你应该仔细考虑，不要因为追求这些效果而带来很大的负面效应，

即使很小的多媒体程序也要占较多的资源，所以只有当确信它们能带来理想的效果时才使用图像和声音。

编写帮助文件时还必须考虑内存的开销。当用户使用帮助时，Windows 帮助系统总是装载所有文件。减少占用内存量的一种方法是使用许多外部文件，即将帮助文件分成许多部分，多媒体都存为外部形式。少量使用图像和声音的目的之一是使你的帮助文件增加趣味性。

建立 Make 文件及编译帮助文件

建立一系列的 RTF 文件可能是编写帮助文件中最费时的工作，与编程时所用的 make 文件（工程文件）不同，编译帮助文件的 make 文件含有的不只是要编译的文件的清单。

本节我们将介绍手工建立 make 文件的方法，Microsoft Help Workshop 程序提供了一种自动的方法我们将在随后的章节中介绍。先介绍手工方法的原因是，使用 HC31——DOS 版的编译程序时必须用它，而且清楚了 make 文件的组成有助于以后调整其结构。

我们从观察一个典型的 make 文件开始。程序列表 15.2 包含了本书中使用的帮助文件的 make 文件。如果用 DOS 程序来编写帮助文件，所有的工作都必须手工填写；不同于我们所看到的 make 文件顶部的注释中的说明，该说明声称帮助编译器自己维护文件，不要开发人员直接修改它们。后面我们将用 Microsoft Help Workshop 程序建立同样的帮助文件。这里我所用到的一些东西并不是典型帮助文件所必须具备的，我会进一步讲解何时用到它们。

注释 程序列表 15.2 中的有些条目因为过长可能被分写成了数行，真正使用这个程序列表时一定要保证每个条目都在一行上。

程序列表 15.2

```
;This file is maintained by HCW.Do not modify this file directly.

[OPTIONS]
HCW=0
COMPRESS=12  Hall Zeck
ERRORLOG=HELP.LOG
LCID=0x409  0x0 0x0 ;English (United States)
REPORT=Yes
CONTENTS=CONTENTS
TITLE=Single Document Application Help File Example
COPYRIGHT=1998  Some Company
HLP=.\Sngl_Doc.hlp

[FILES]
.\Sample.rtf
.\Glossary.rtf
```

```
[MAP]
#include ..\Resource\my.hm
CONTENTS=1          ;Main help file menu.
FORMAT_TOOLBAR=2        ;Format toolbar explanation shortcut.
GLOSSARY=3              ;Glossary window shortcut.
HID_FORMAT_FONT=0x1E160     ;Added for ID_FORMAT_FONT in AFXRes.h file.

[WINDOWS]
Main="A Sample Help File",(0,0,800,600),52484,(rl4876671),(r12632256),f7;This is the main window.
Glossary="A Sample Help File - Glossary",(50,50,850,650),52848,(rl4876671),(r12632256),f7; The direct Glossary help window.

[CONFIG]
CB("glossary","&Glossary","JI('SNGL_DOC.HLP>glossary','GLOSSARY' )")      ;Add a Glossary  button to the display.
CB("controls","C&ontrols","JI('SNGL_DOC.HLP>main','FORMAT_TOOLBAR')")      ;; Add a jump to the controls bitmap button.
RegisterRoutine("USER","EnableMenuItem","uuu")    ;Enable (or disable) a menu item.
RegisterRoutine("USER","GetSubMenu","u=uu")    ;Get the name of a submenu.
```

```
RegisterRoutine("USER","GetMenu","u=u")     ;Get a menu name.
RegisterRoutine("USER","GetActiveWindow","u=")     ;Get the active window name.
RegisterRoutine("USER","DrawMenuBar","u")     ;Instruct Windows Help to draw a menu bar.
EnableMenuItem(GetSubMenu(GetMenu(GetActiveWindow()),1),0,1027)          ;Disable the copy option of the Edit Menu.
DrawMenuBar(GetActiveWindow())          ;Redraw the menu when we're through.
```

正如你所看到的，这个 make 文件看上去有些庞大，我们最好分节来解释。例如，观察 FILES 一节，它列出的是脚本文件（按微软的叫法是“主题”）清单，我们先仔细讲解一下第一小节：

Make 文件的 OPTIONS 小节指示帮助编译器如何编译文件。这里的大部分条目都是不言自明的（当然是对英文好的人士来说——译者注）。例如：COPYRIGHT 语句出现在帮助窗口的“关于...”对话框，COMPRESS 语句指示编译器是否对文件进行压缩以及如何压缩，HCW 语句是 Microsoft Help Workshop 专有的，使用 DOS 编译器时用不到。最需要注意的是 HLP 条目，帮助文件的文件名和 Visual C++程序名相同，虽然这不是必须的要求，但使用这样的名称却会极大地简化以后的操作，MFC 会自动完成调用与程序同名的帮助，当然二者的扩展名是不同的。

MAP 小节用于设置帮助内容的属性，在此出现的每个词都必须是脚本文件中的某个帮助主题标识符（#脚注所标出的内容）。为每个跳转赋值实现了从控件中访问帮助主题的功能。用户只需选中控件按 F1 就可以看到关于该控件的帮

助。注意：我们没有把程序中用到的所有控件都罗列出来，而只是包括了 HM 文件，其好处是不仅节省了录入时间，还有助于发现帮助文件中的漏洞。

有多种方法显示帮助文件中的数据，我一般为每个主题开设一个窗口。在 make 文件的 WINDOWS 小节列出了两个窗口：一个是主窗口，这个窗口总是显示的；另一个是词汇表窗口，当用户点击工具栏上的“Glossary”按钮时就会显示该窗口，通过它用户可以在不失去当前位置的前提下查找单词。显然并不是每次用户访问词汇表时都显示这个窗口，如果用户点击帮助文件中的超链接文本访问词汇表，词汇表就显示在主窗口中而不会另开一个窗口，这时用户只需用“返回”按钮就可以返回到以前的位置。

CONFIG 一节是我要着重介绍的，这是建立 make 文件时费时最多的部分，原因也正在于它最灵活。我的 make 文件中有三类事件：建立按钮、注册函数及执行一系列的 Windows API 调用。显然，可以加入任何事件，我们先看一看下面这个相对简单的例子：

由建立两个新按钮开始。第一个是“词汇表”按钮，这时要用到两个不同的宏调用，一个用于建立按钮，以“Glossary”作为它的标签名并在字母“G”下加下划线，功能是跳转到 JI 宏调用所返回的标识符位置，JI 宏调用看上去使用了两个参数，实际使用了三个参数：“SNGL_DOC.HLP>glossary”实际提供了两条信息：一是告诉 JI 宏使用哪个帮助文件，二是告诉 JI 宏使用哪个窗口显示帮助主题。另一个参数提供主题标识符的名称——请记住，这是一个#型脚注。

接下来要做的工作是向 Windows Help 注册一些 DLL 函数。注册的函数数目不受限制，注册过程分为三步：第一，告诉 Windows Help 调用哪个 DLL，

Windows Help 总是假设 DLL 在 SYSTEM 目录下，所以要么把 DLL 放在这个目录下，要么给出它所在的路径。我一般都是放在 SYSTEM 目录下，因为无法预测用户将来会把它放在哪里。第二，给出该 DLL 中所用的函数名字，注意大小写要与原函数完全一致——过去我曾遇到过因函数名不一致带来的麻烦。第三，告诉 Windows Help 该函数将要查找何种参数。当我注册 GetSubMenu 函数时需使用“u=uu”，即：此函数需要两个无符号数作为输入，返回一个无符号数作输出。共有四种数据类型：无符号数（u），带符号数（i），字符串（s）或未知类型（v）。等号（＝）总是用于分隔输入值与输出值。

在注册过 API 调用之后，可以使帮助的显示发生一些变化（可以做的包括所有注册过的函数提供的功能）。以禁止使用 Edit|Copy 命令为例，虽然还可以看到它，但已变为灰色的，即不可执行。使用这种技术可以使帮助做的非常时髦，注意完成时要用 DrawMenuBar 函数来重画菜单条，否则所做的修改可能显示不出来。

最后一步就是编译帮助文件了。如果用的是 DOS 版的帮助编译器（HC31.EXE），必须在[CONFIG]小节给出记录文件（log 文件），我的 make 文件中用 ERRORLOG=HELP.LOG 语句来实现，使我可以用文本文件来查看帮助文件中的错误。不便的是，DOS 版的帮助编译器是命令行驱动的，界面不够友好，如果不指定记录文件，你就只好期望有一目十行的本事了。所要做的很简单，键入：HC31<MAKEFILE>.HPJ，make 文件一定要带 HPJ 扩展名，否则编译器找不到。在建立过程中，编译器会在屏幕上显示许多“……”，当再次显示 DOS 提示符时，表示编译已经完成了。

15.4 使用 Microsoft Help Workshop

早期的 Windows 程序设计人员即使编译一个很小的工程文件也要学习很多东西，那时的工具很简陋，即使一个小程序也要编很多代码。那时是 DOS 的天下，但随着编译程序的改进，DOS 已是昔日黄花；但编译帮助文件时还是要用到那些编译程序。实际上我也很反感用 DOS 程序来建立帮助文件，但是在最新版的 Windows95/98 编译程序出现以前也只能使用它。即使是第三方厂家专门开发的编译程序有时也要和晦涩的 HC31 命令行程序打交道，它们在编译过程的可视性方面并没有实质性的改进。

注释 本节专门讨论了 Windows 帮助文件，如果关心 HTML 帮助文件的建立技术，请阅读后续章节。

在引入了 Microsoft Help Workshop 之后，编译过程有了全面的改观。再也不用依赖于基于 DOS 的编译程序了，Microsoft Help Workshop 在 Windows 环境下建立并编译帮助工程文件（帮助 make 文件的另一种叫法）。图 15.3 是 Microsoft Help Workshop 典型的显示示例。

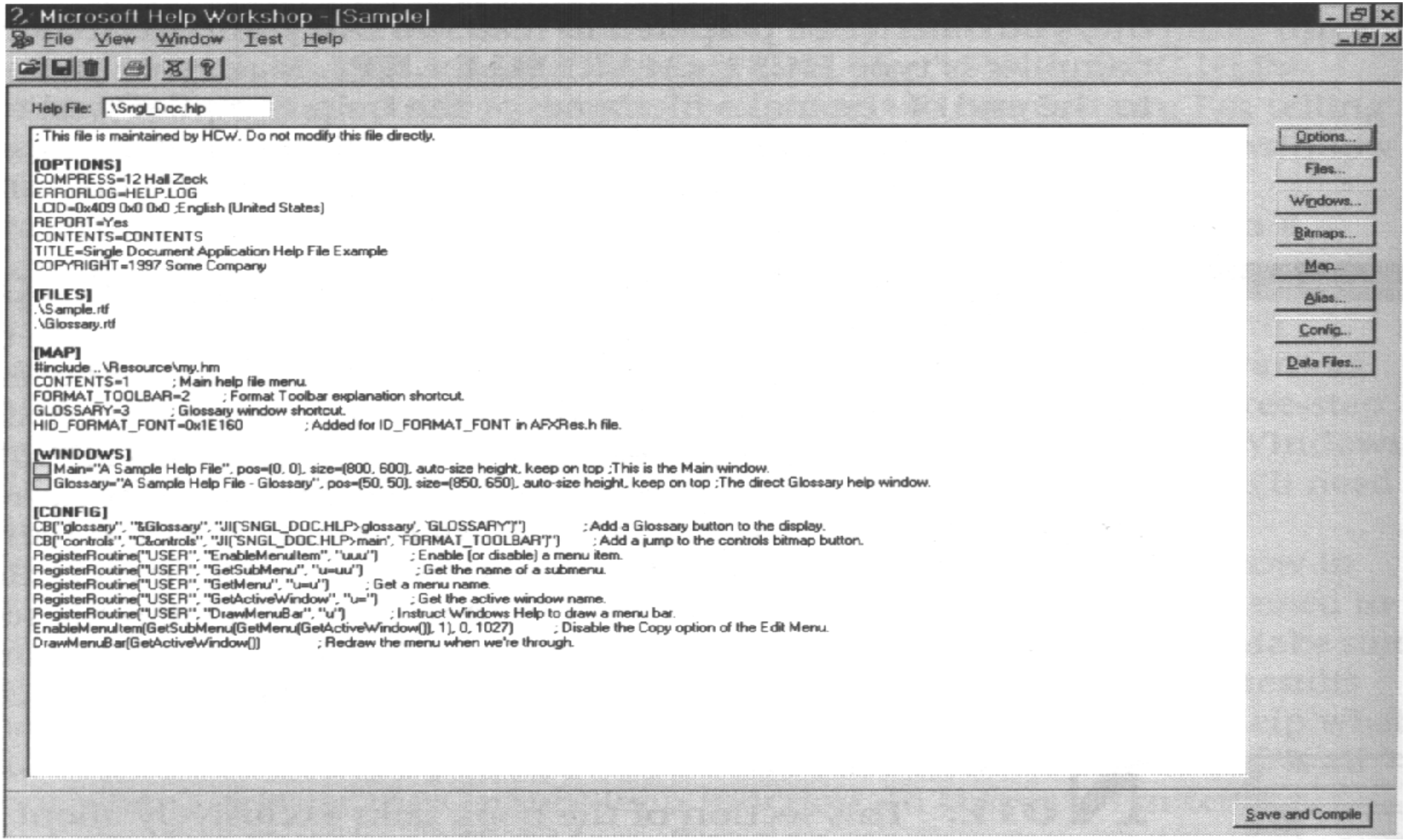
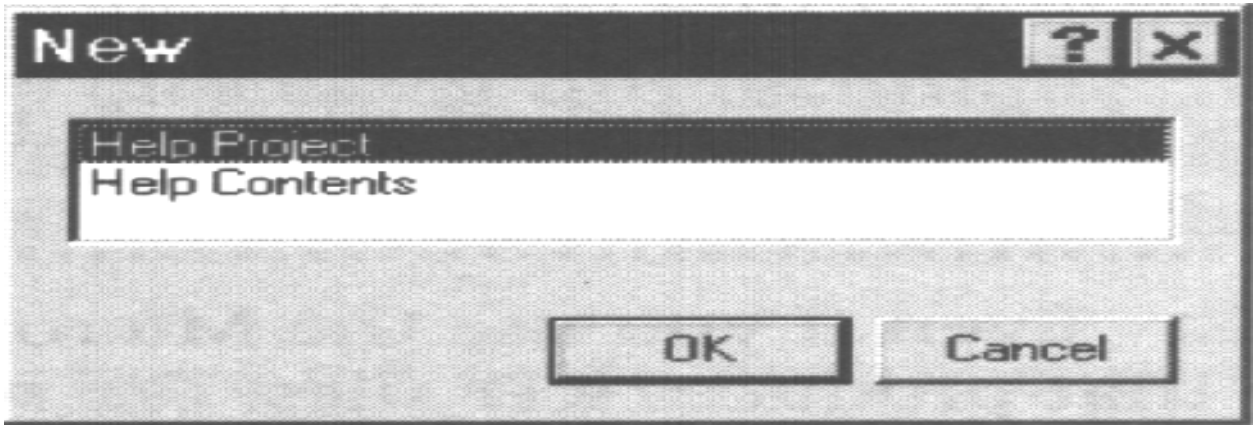


图 15.3　Microsoft Help Workshop 提供了一个方便的 GUI 用以建立帮助文件

注释 即使使用 Microsoft Help Workshop，也还是需要一个生成 RTF 文件的编辑器。我不知道 Microsoft 为什么没有改进这一方面，事实就是如此。我还是用 Microsoft Word 建立 RTF 文件，在使用 Microsoft Help Workshop 建立帮助文件时，所用的 make 文件指向这些文件。

我们看看这个工具如何降低建立过程的复杂性，所建的 make 文件如程序列表 15.2 所示（假定已经建立了包含帮助脚本的 RTF 文件）。第一步是建立一个新的工程文件，用 File|New 命令实现，系统会显示如下图所示的对话框：

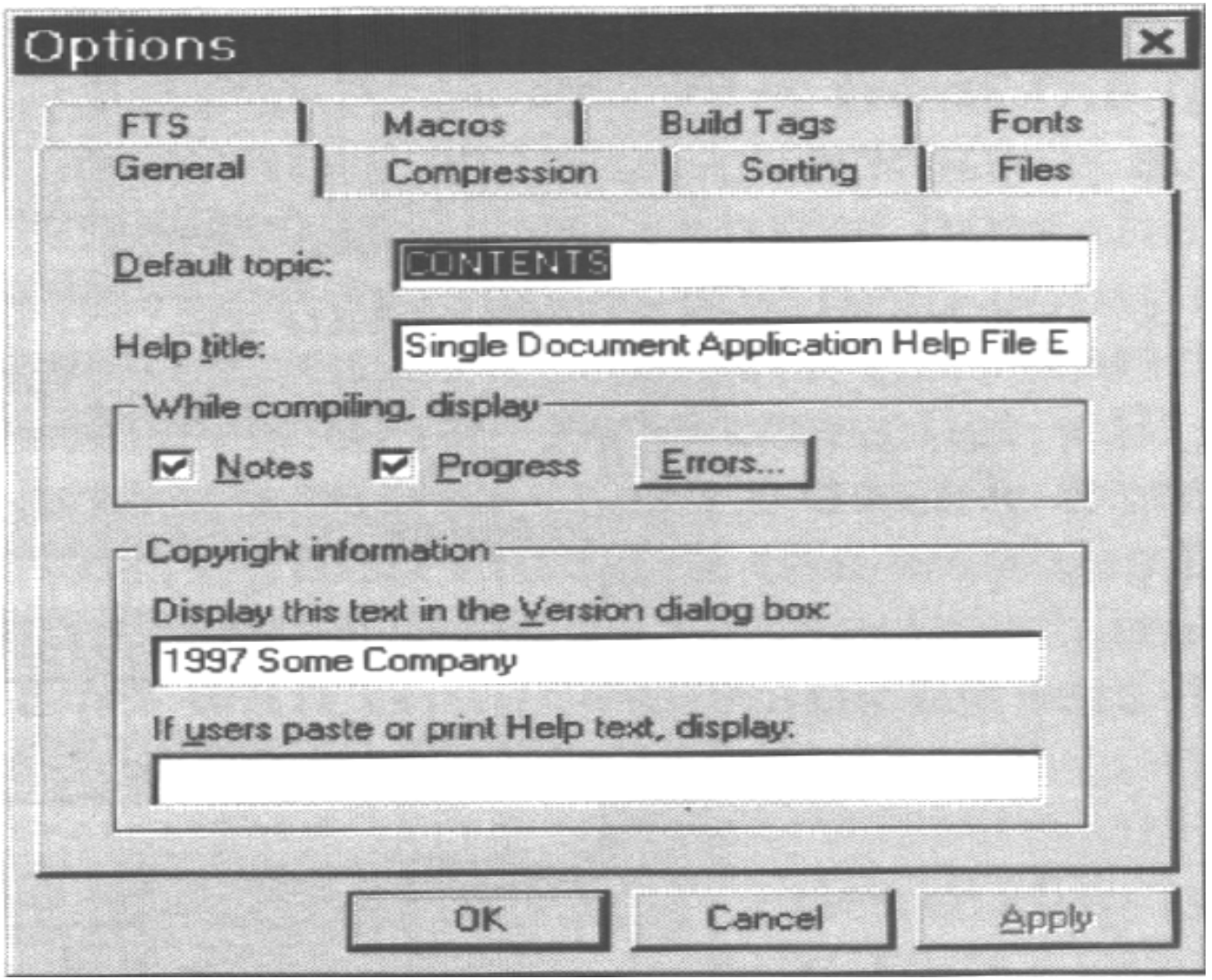


选择 Help Project 后，按 OK 完成。如图 15.3 所示，已经建成了一个新的帮助工程文件，它初始为一个空白文件，由你填入帮助文件的特性。

警告 用 Microsoft Help Workshop 建立的文件与用 DOS 编译程序建立的文件互不兼容，必须预先做好决定：新的 Microsoft Help Workshop 对开发者更好用，但只限于 32 位的操作系统环境；如果还需要对 Windows 3.x 向下兼容，就只好用 DOS 编译程序了。

定义工程的各种选项

当开始使用一个新的工程文件时，总要先定义一些选项。例如：至少要知道调用帮助文件的目的，要加进哪些版权信息。我一般以内容主题为主目录，所以把它加在帮助文件的开头位置是适宜的。点击 Options 按钮，系统就会显示 Options 对话框：

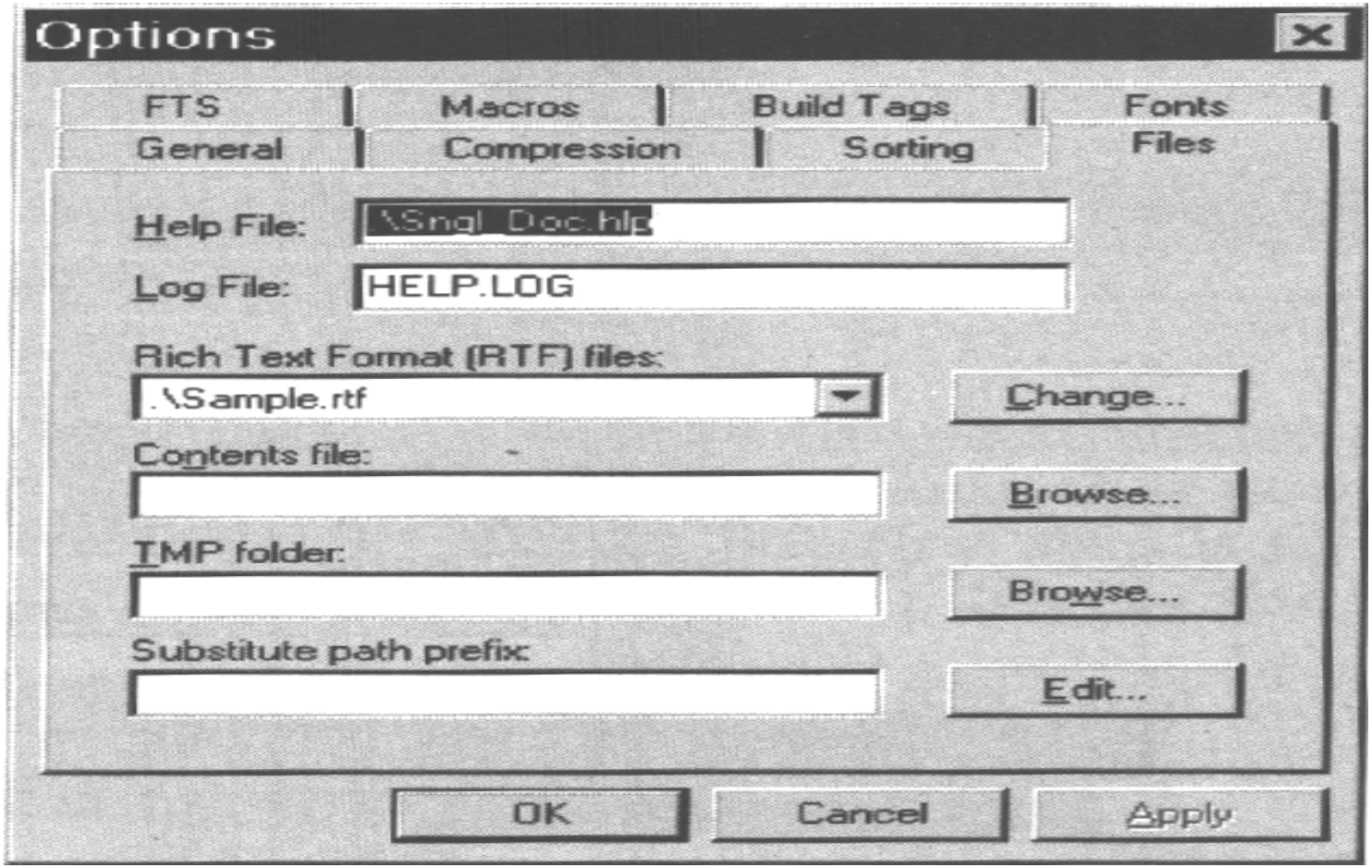


注意此时已经定义了一些基本选项，这些选项的含义类似于前面章节中 make 文件描述的内容（你会看到 Compression 属性页上的选项相当熟悉，它们也是用来压缩帮助文件大小的）。此时，有些人会觉得这个工具并没有什么优越性，特别是当他们的盘上有很多预定义文件时。要指出的是这个对话框使你只需填

填表，而不用再一条一条地完成语句。

Options 对话框的“Sorting”属性页有二个区域，第一个区域决定了帮助文件使用的语言，语言决定了排序原则，每种语言的字母顺序都不相同。第二个区域有两个选项，一个选项允许忽略非空格字符，例如：如果不选此项，出现在 ê 上的（ ^ ）就会影响排序；另一个选项表示排序过程忽略帮助文件中的所有符号项，这对于为数据录入程序或其它一般应用程序建立非特殊的索引时非常方便。另一方面，为参考手册型的帮助文件建立索引时它又确实很碍事，因为许多的 C 函数都是以“_”开头的，忽略“_”就不容易识别函数。

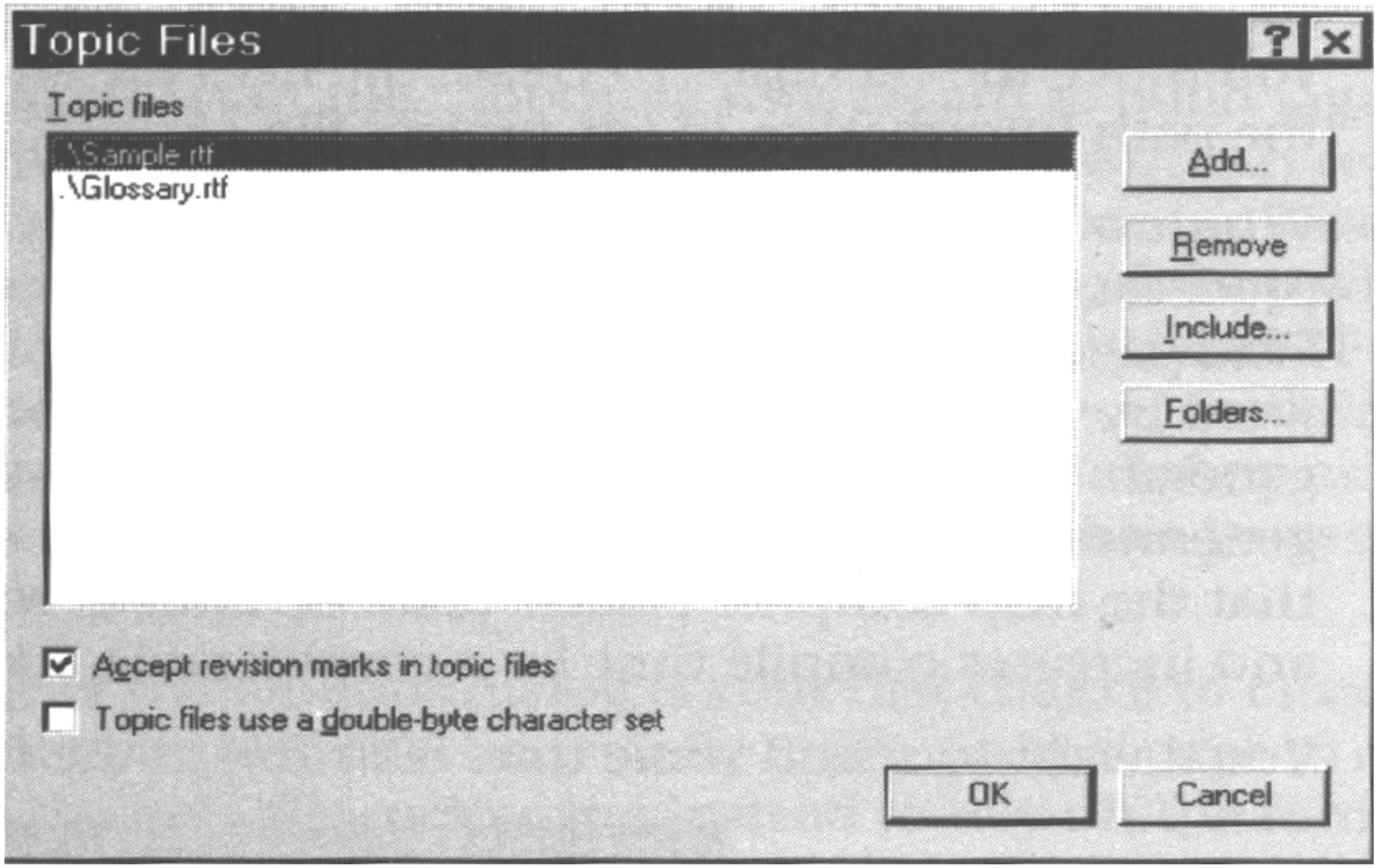
Options 对话框的“Files”属性页如下图所示：



你会注意到它的许多选项我们以前都介绍过，修改 Help File 域可以给帮助文件改名，一般帮助编译器用工程文件名作为帮助文件名的基础；Log File 域包含了记录文件的文件名，所幸的是这一项在新的帮助编译器中已不是必需的了。我还是用它来记录编译过程，但现在这是可做可不做的事情。

在此属性页上的最重要的域是 RTF 文件清单框，当前帮助文件的工程文件

所含的文件都列在这里，点击“Change”按钮显示“Topic Files”对话框，如下图所示：



从这里可以增加或删除工程文件中 FILES 小节所列出的主题文件。注意此对话框中的两个复选框，它们很重要，因为它们控制着帮助编译器如何响应 RTF 文件。第一个选项允许编译器在下一次编译时自动执行对 RTF 所做的更改，如

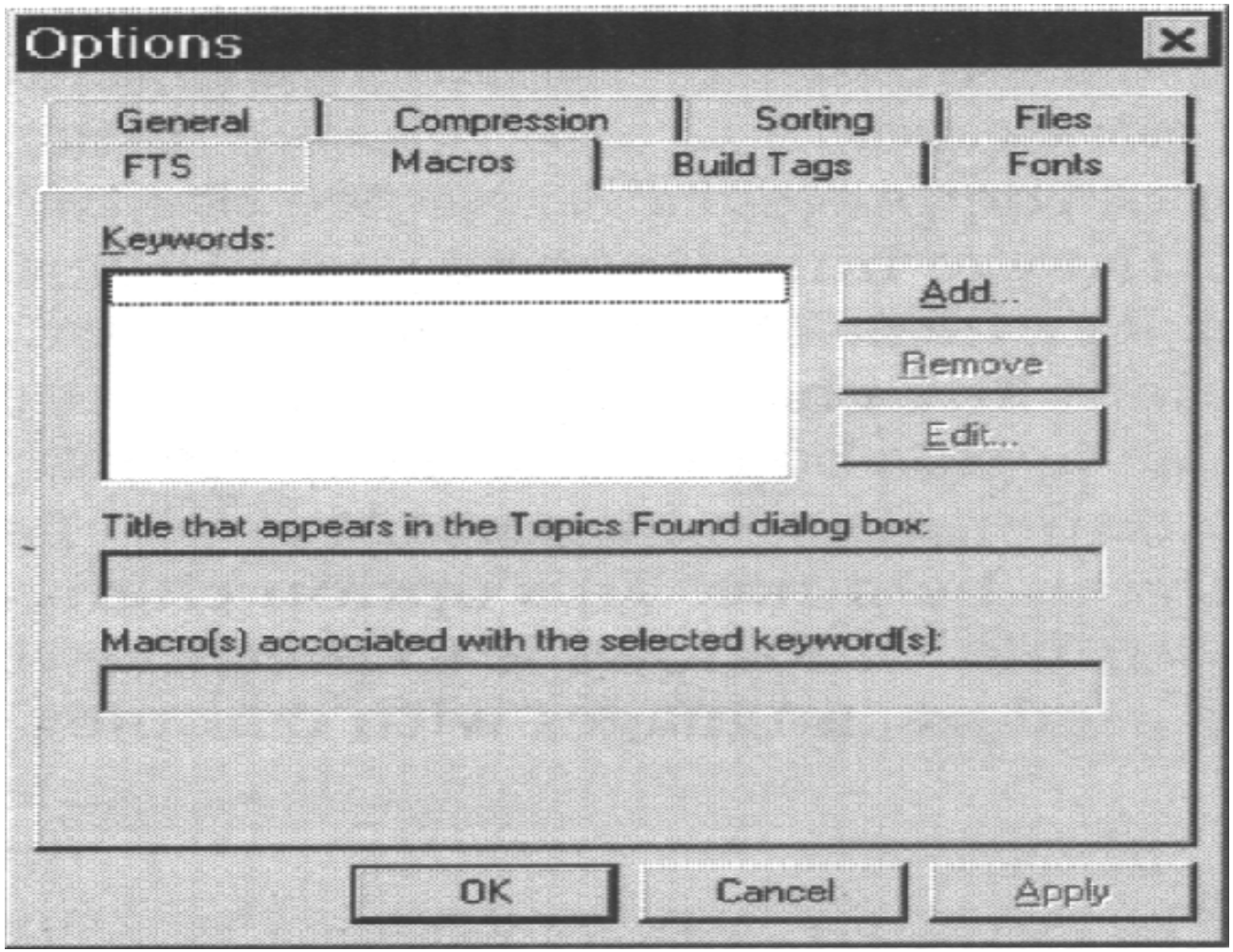
果未选中该项，编译器就会忽略所做的改动。第二个选项对使用双字节字符集（DBCS）的帮助文件非常重要，它改变了帮助编译器的工作方式，从而保留了特殊字符（该特征主要用于特殊的语言，比如中文等）。

技巧 进入“Topic Files”对话框的另一途径是单击主窗口的“files”按钮。

在“Options”菜单的“Files”属性页还有其它一些选项，其中之一就是“Contents File”域，如果是新建一个工程文件，Help Workshop 在建立内容页时就会自动地填充该项。选中该选项的原因是已经有了一个内容页又想用于当前的工程文件中。当帮助文件超过 8MB 时要用到 TMP Folder 选项，它给出了 Help Workshop 编译时建立临时文件的目录，一般情况下用不到，只是当缺省目录的磁盘空间不足时才用到。最后一个域是“Substitute Path Prefix”，当移动了某些文件又不想改变工程文件中的路径信息时要用到。

Windows 95/98 的帮助文件提供了一些以前没有的功能：全文查找。当选中“帮助主题”对话框的“查找”页就会建立这个数据库，实现对全文的逐字查找。编译过程中“选项”对话框的 FTS 页中有一个选项用于生成这个文件，由于 Windows 95/98 总是要生成它，一般都不用选中。编译器建立的 GID 文件要在发行盘上占用很多空间并增加了编译时间。

下面有必要学习一下“Macros”属性页，其图像如下图所示：



从这里定义关键字宏供全文使用。而且当用户查找某一主题时，这些宏会出现在帮助主题对话框的索引页中。

单击“Macros”属性页的“Add”按钮，系统显示一个含有三个域的“Keyword Macors”对话框，第一个域含有宏名字，第二个域指示宏本身，第三个域指示

Help Workshop 如何在索引页显示该宏，它用于有多个帮助文件但只想在某个选定文件范围显示关键字的情况。例如：一般把词汇表和程序列表名的缩写集中存放在一个独立的文件中，用 JI 宏建立一个全文范围的跳转。就用关键字宏的方法来实现，用户根本不会感觉到又加载了其它的文件——因而对用户是透明的。

注 请谨记，Options 对话框的 Macros 属性页的设置会影响整个帮助文件，而不只是一个窗口。

前面我们介绍过“*”脚注用于建立标识，Options 对话框的 Build Tags 属性页就该用到它了，前面对这个脚注已经讲得十分透彻，这里不再赘述。中心思想就是给 Help Workshop 提供帮助文件要包括的建立标识的一个清单。即使 RTF 文件含有主题，如果不做主题标识，该主题也不能出现在帮助文件中。如果没有填写该属性页，Help Workshop 会包括所有 RTF 文件中的所有主题。

Options 对话框的 Fonts 页是第一个用于定制帮助文件外观的地方。我一般不用，而是在字处理过程中就完成了。然而，如果用文本编辑器建立了一个 RTF 文件，再用该选项可以节省时间。Character Set 域允许选择字符集，缺省为 ANSI，可以选择其它的语言，如阿拉伯语。在 WinHelp Dialog Boxes 域的 Font 用于指定缺省字体，单击 Change 按钮，显示含有三个域的 Font 对话框。第一个定义字体名称，第二个定义字体大小，第三个定义字符集。在 WinHelp Dialog Boxes 域的 Font 下拉列表框用于更改 Windows 帮助文件的通用字体，允许用一种字体代替另一种字体，Add 按钮显示一个 Add/Edit Font Mapping 对话框，它含有两个组，每个组有三个域。这三个域与前面描述的 Font 对话框中的相应域完全

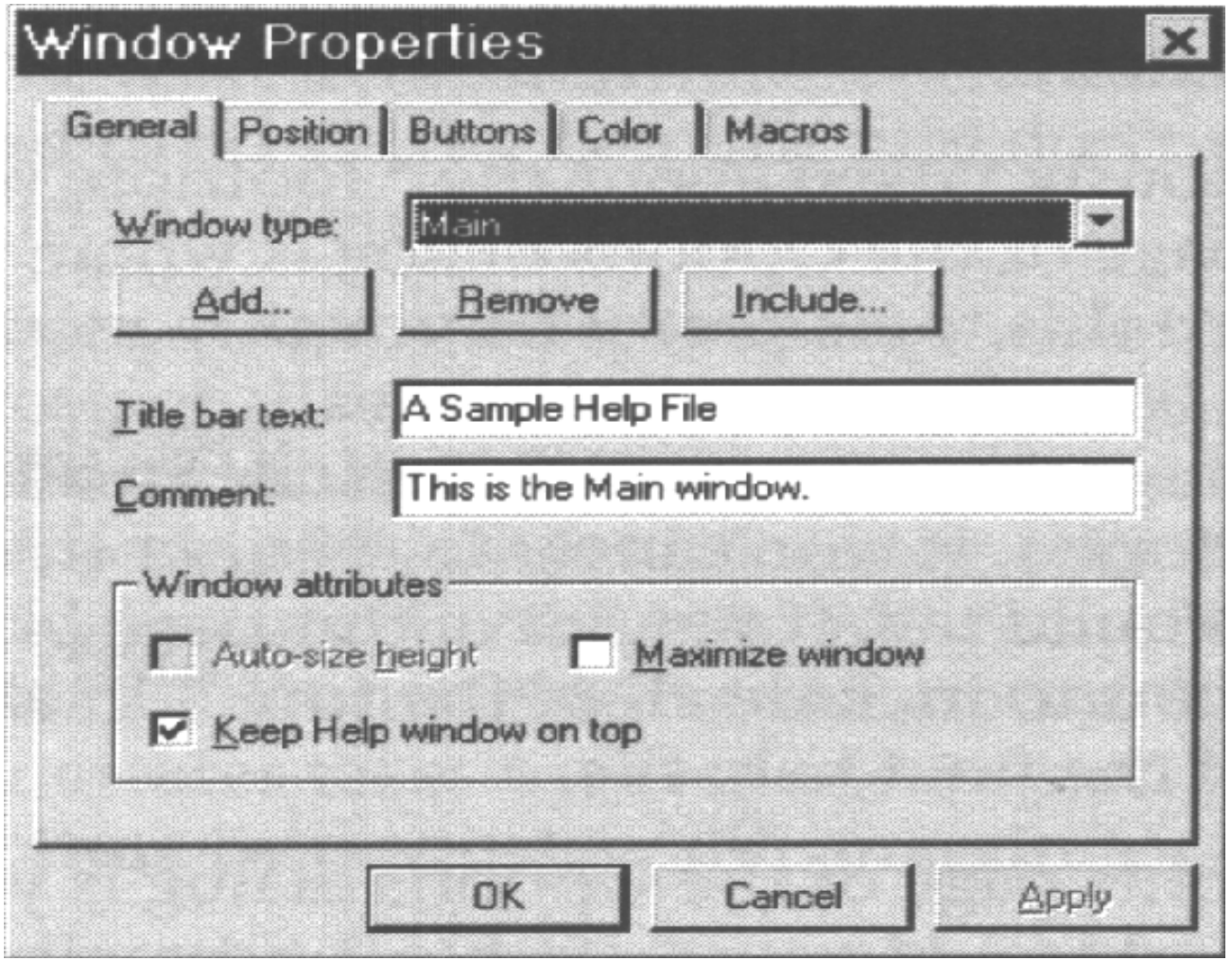
一致，唯一的问题是如果字处理程序覆盖了这些设置，那么它们就不起作用了——这是使用像 Word for Windows 这样的产品编辑文件时经常发生的事。

定义窗口

定义窗口选项只是建立工程文件的第一个步骤。当这些选项起作用时，需要定义一些窗口来显示数据。一般建立一个叫 Main 的窗口，它是应用程序用到的主窗口。

建立窗口很简单，只需单击 Main 窗口的 Windows 按钮（图 15.3），系统就会显示窗口属性对话框：

首先应关注的是 General 属性页。单击 Add 按钮，系统显示一个带有两个域的 Add a New Window 对话框，一个域是窗口的名称，另一个域指明窗口类型，Help Workshop 可以建立三种类型窗口：过程窗口、引用窗口以及出错消息窗口。过程窗口和引用窗口的差异很小，它们都可以自动调整大小，并且都包含了三个系统按钮。这两类窗口的最大差别是它们在屏幕上的放置位置——后面我们将会讲到如何设置位置信息。出错消息窗口不同于其它两个窗口，它并不包含三个系统按钮，而更像一个对话框。

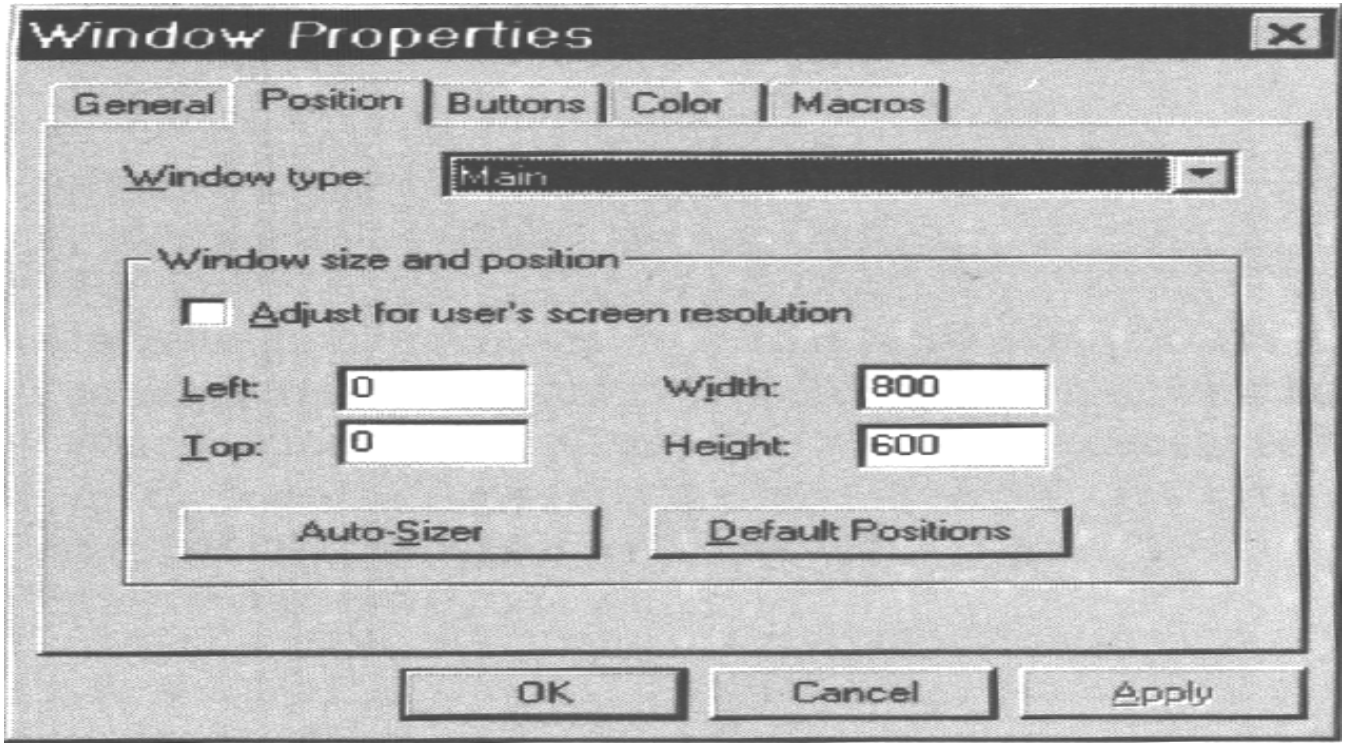


返回到General页，Title Bar Text域定义了Windows Help在标题栏里放置什么，它不影响帮助窗口的标题区的实际显示。Comment域是在工程文件中加入注释，我总要在工程文件中加入注释，以帮助随后理解帮助文件的建立过程。该属性页中还有三个属性复选框，Help Workshop可以根据当前情况禁止其中一

个或多个复选框，例如：你不能让主窗口自动调整大小，否则当允许某个辅助窗口自动调整大小时，该窗口打开后，用户就无法使其最大化。大部分过程窗口缺省情况下总显示在屏幕的最外层，如果你想让操作你的程序的用户随时都可以看到帮助，这就是实现该功能的一个便利的特性。

技巧 为了更好地控制窗口显示的外观，我总是关闭 Auto-Size Height 特性。Position 属性页的选项将对帮助窗口提供全面的控制，后面我们会具体介绍该属性页。

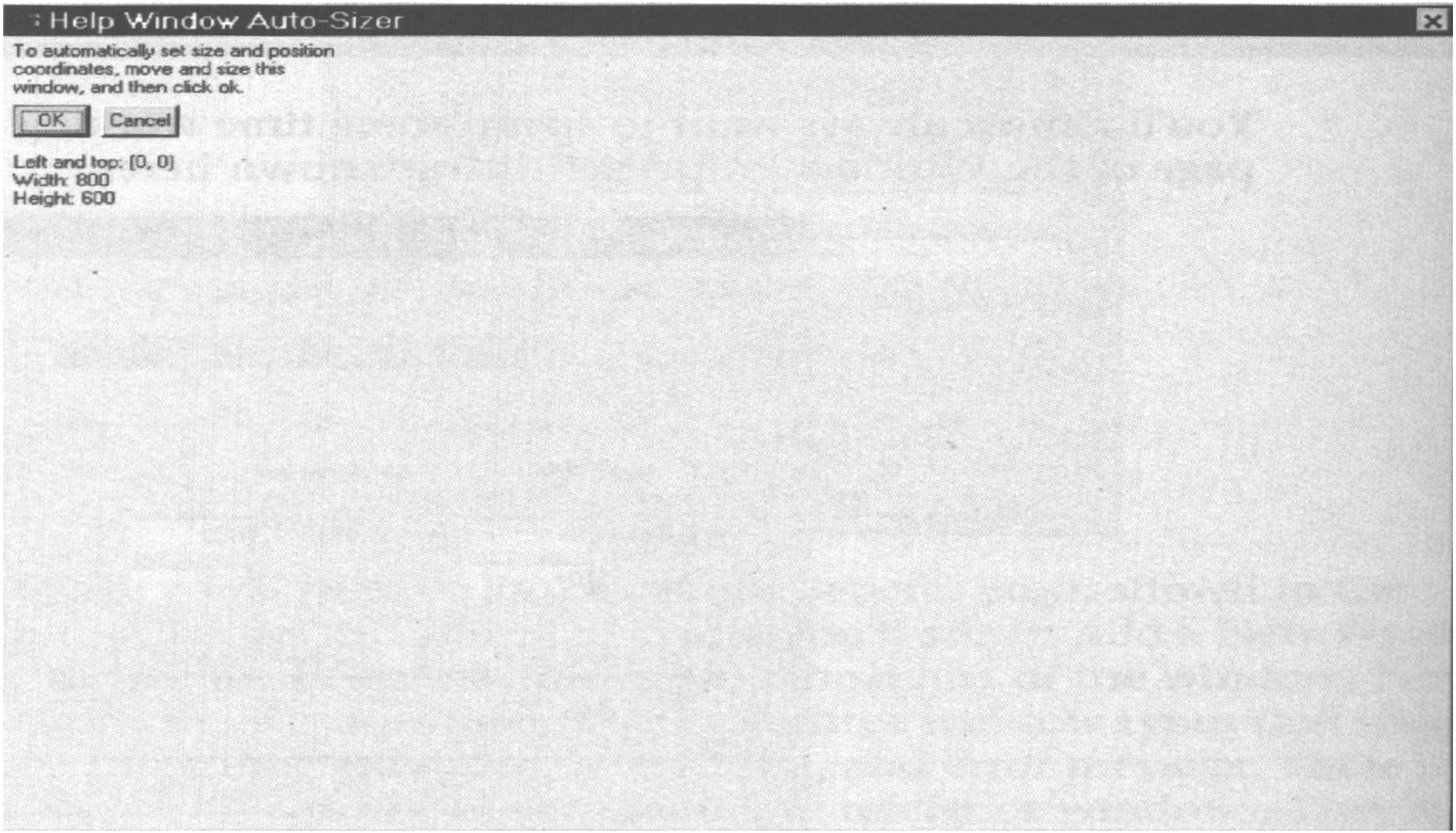
你可能会经常调整 Windows Properties 对话框的 Position 属性页，该页如下图所示：



它比其名称所包含的功能要强一些，该属性页除了能控制帮助窗口的位置及大小外，还提供了很多非常容易使用的特性。

在这个属性页上有四个域：Left、Top、Width 和 Height，它们控制窗口的大小和位置。一般第一个帮助窗口放在左上角，大小为 640×480 或 800×600，具体使用哪一个值要随机器的性能而定。这样的窗口看上去可能会小一点，但用户一般都会自行调整。如果程序员将窗口放得离边缘太近，使用时要想在用过新窗口后又返回老窗口时就很不方便。好在还有 User's Screen Resolution 选项，当用户的显示器档次太低时，用该选项可以防止帮助窗口被全部隐藏起来。

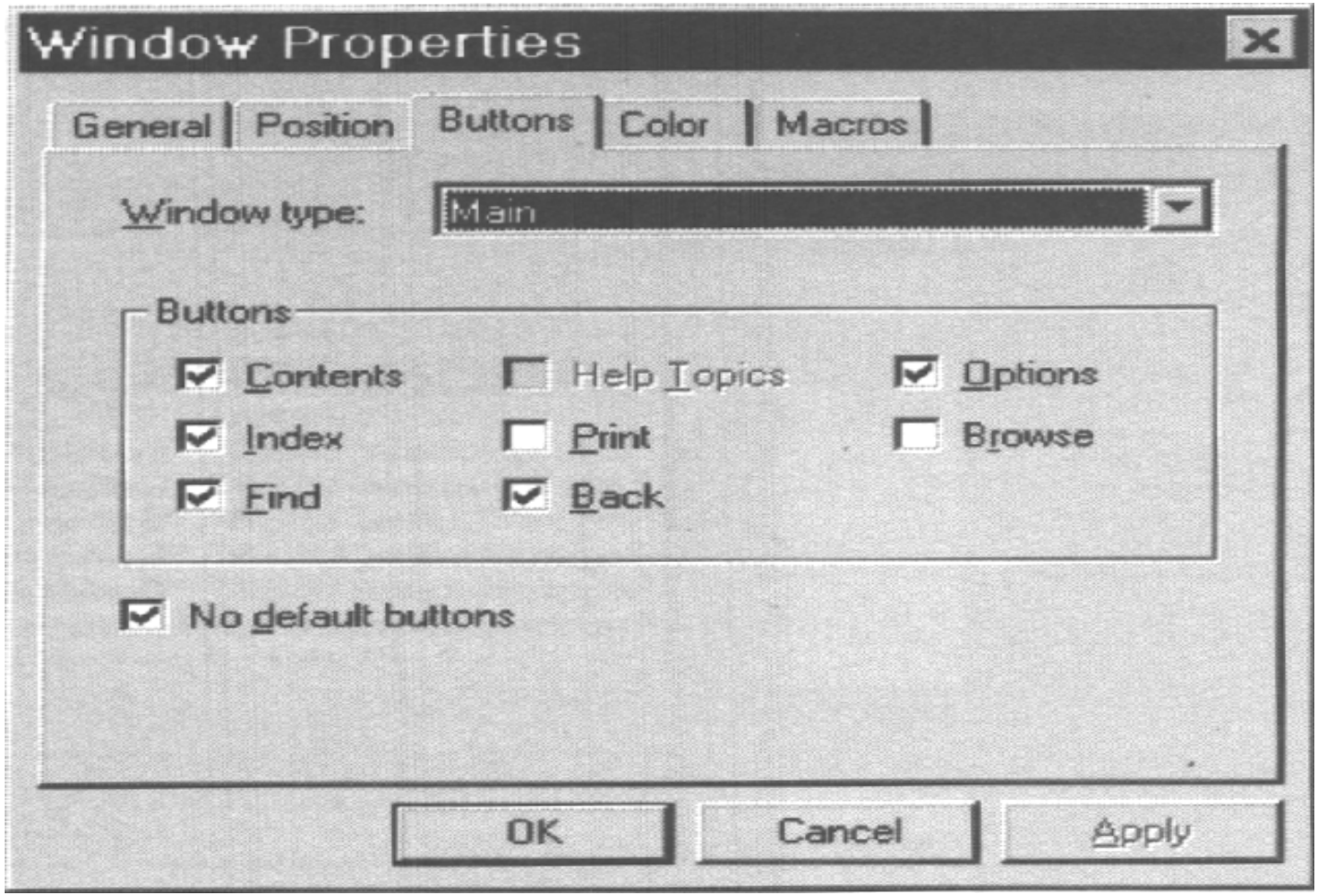
该属性页还有一个非常特别之处，起初你可能会不注意，看一下 Auto-Size 按钮，单击它会显示如下图所示的窗口：



如果改动了窗口的位置，Left 和 Top 域的值也随之而变；重新改变窗口的

大小则会改变 Width 和 Height 域的值。这种用图像方式改变窗口的方法减少了重编译帮助文件的次数。

Windows 95/98 定义了许多可以加入帮助文件的缺省按钮，有时可能不会都用到，例如：如果在 RTF 文件中未定义浏览（+脚注）功能，那么 Browser 按钮就用不到了。这里显示的 Buttons 页用于定义帮助窗口中用到的按钮。

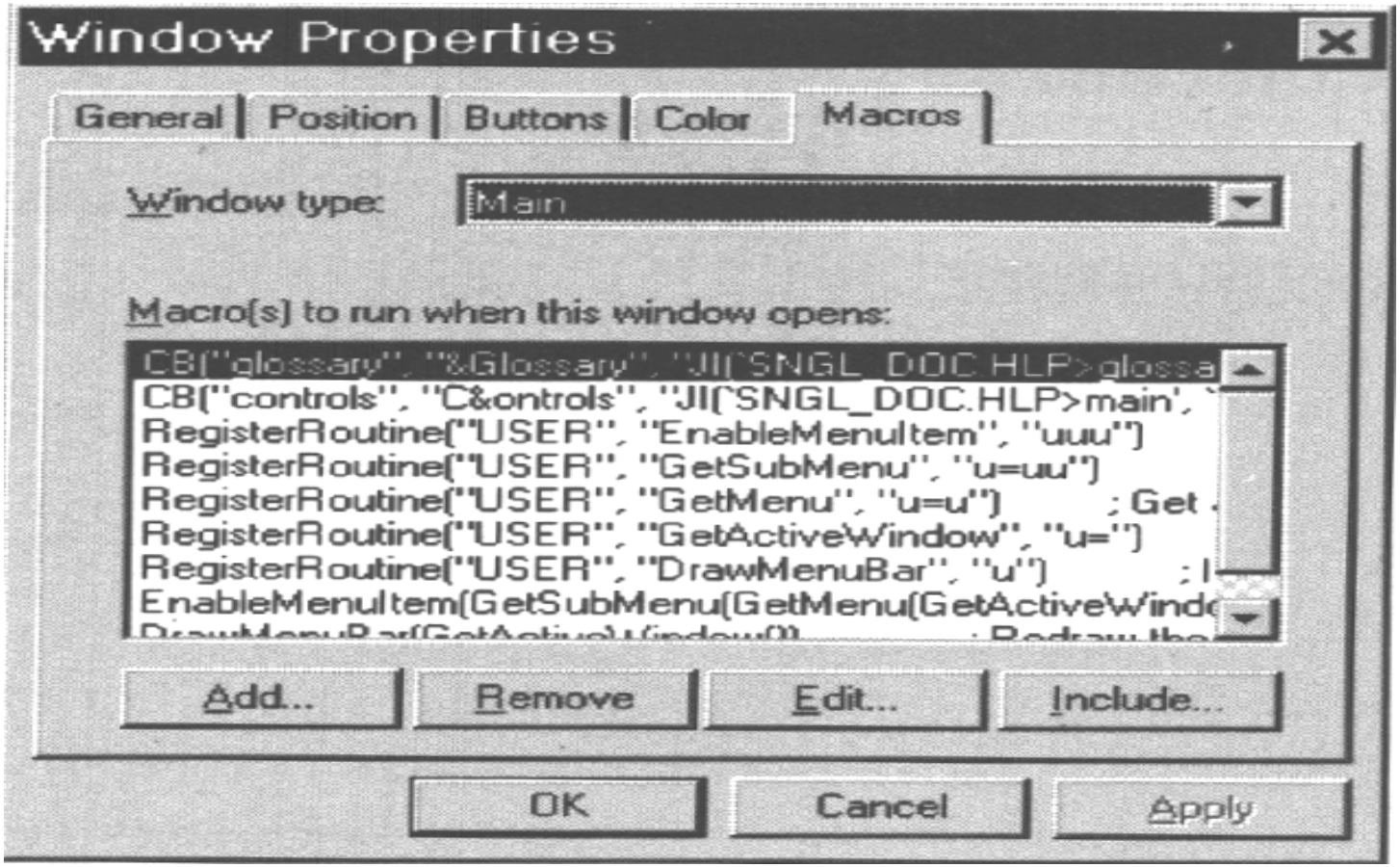


所有辅助的过程窗口和引用窗口都用不到 Contents 和 Index 按钮，主窗口中则把它和 Print 及 Back 按钮一起作为缺省按钮提供；另一方面，主窗口不用 Help Topics 按钮。与所有其它类型的窗口不同，出错信息窗口没有任何限制，可以随心所欲地使用各种按钮。

技巧 单击 No Default Buttons 复选框，可以不受 Help Workshop 在主帮助窗口按钮设置方面的限制。这个复选框仅在主窗口中出现，即不能修改辅助性过程窗口和引用窗口中存在的限制。

下面介绍 Color 属性页，它有两个域：Nonscrolling Area Color 和 Topic Area Color。每个域都有一个 Change 按钮。单击它会出现一个调色板，从中选择不同的颜色，帮助窗口的颜色也会随之而变。

最后一个属性页是 Macros 属性页，它的外观如下图所示：



主窗口总是使用工程文件中 CONFIG 节的宏作为缺省宏，本节中你看到的所有宏都是可自我执行的——这就是 CONFIG 节的宏都加到主窗口的原因。你希望打开主窗口时让那些宏运行。在主窗口中插入新宏就是靠向工程文件的 CONFIG 节中加入宏来实现的。向其它窗口中加入宏会改变那些窗口的显示，例如：如果向一个辅助性窗口中加入浏览功能，需要加入一个或多个宏以设置一定的条件，从而忽略在 Buttons 属性页的 Browse 按钮上所作的设置。这些辅

助性窗口在帮助工程文件中要专门设置 CONFIG＜窗口名＞小节。

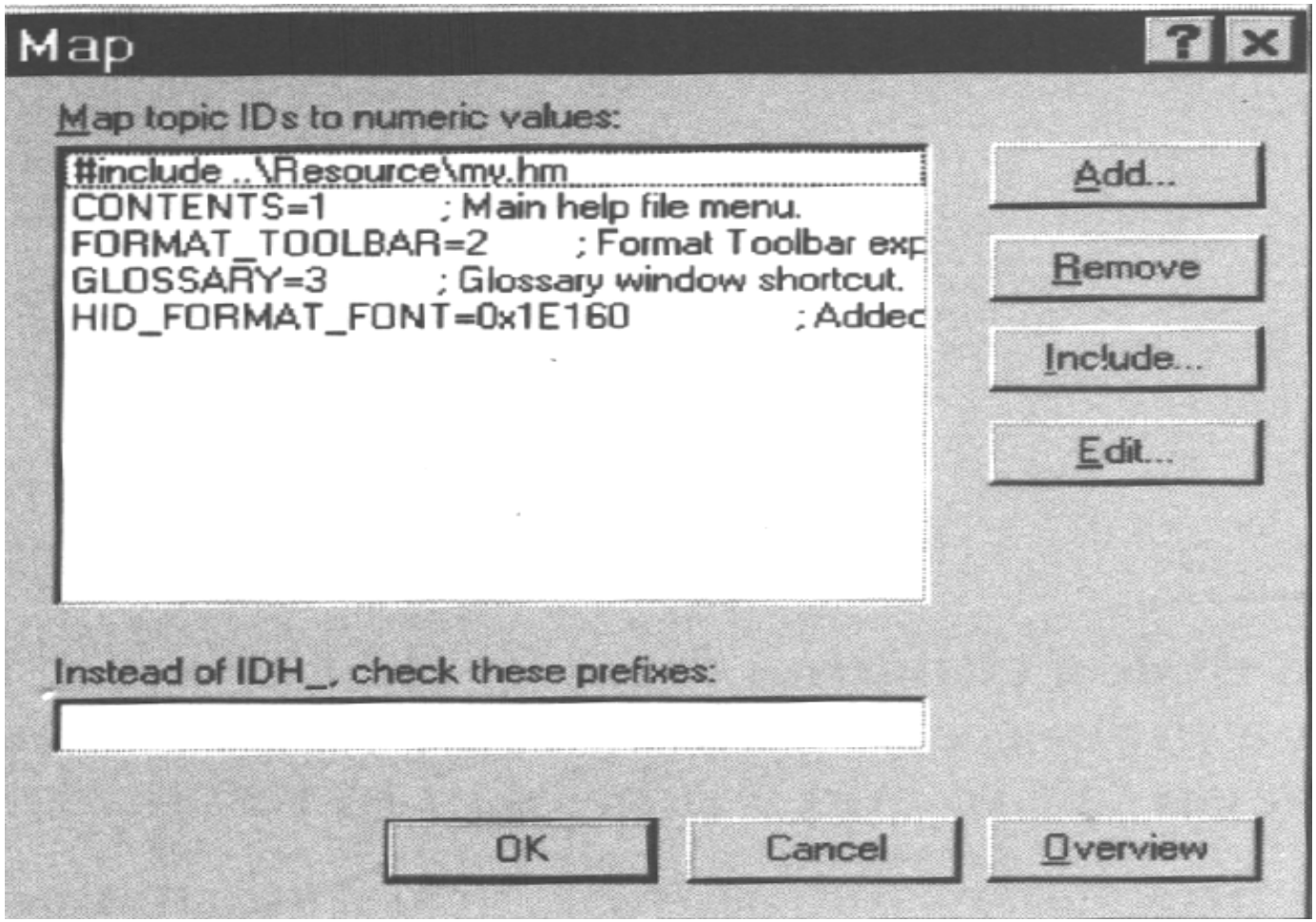
注 添加到 Windows Properties 对话框的 Macros 属性页中的宏通常只影响一个窗口，而不是帮助文件中的所有窗口。

技巧 访问主窗口 Macros 对话框的另一种方法是单击如图 15.2 所示的主窗口的 Config 按钮。

映射帮助主题

前面已经陈述过这部分的重要性，如果没有把帮助文件中的主题标识符映射到帮助内容序号上，就不能建立一个对应用程序中控件内容上下文相关的帮助文件。下面我们将讲述如何实现这种功能。

单击 Map 按钮，系统显示如下图所示的 Map 对话框。



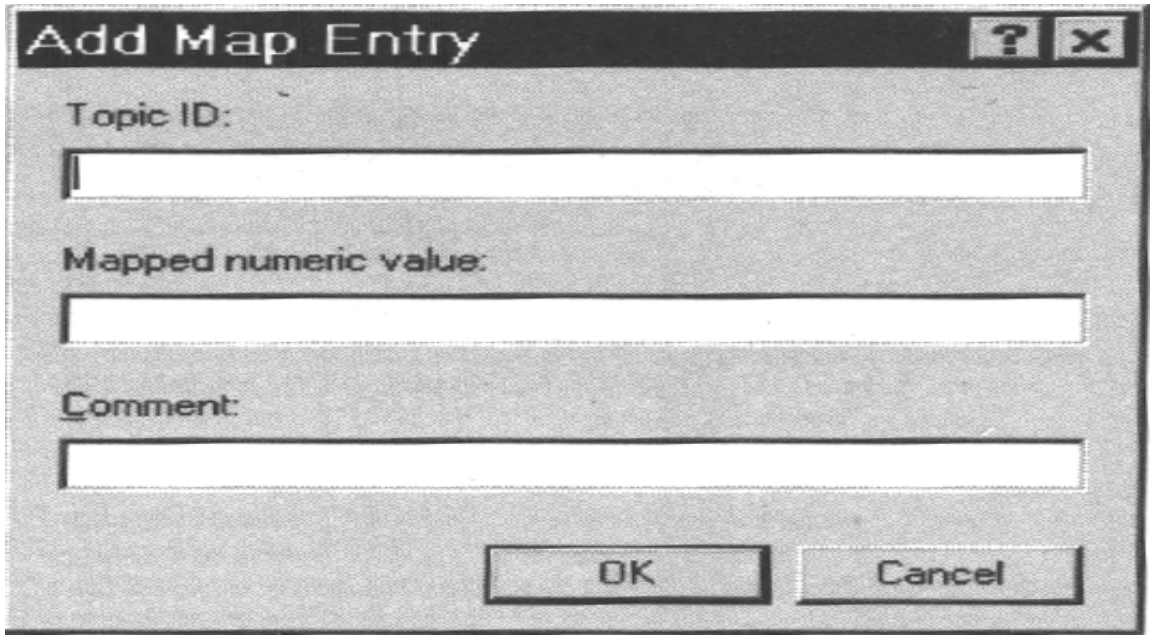
在这里可以定义主题标识符与内容序号间的关系。上图中，我已经定义了一些联系，主题标识符被设置为与帮助内容序号相同的值，紧跟其后的是描述这一项目的注释。

有许多方法可以直接保持内容序号，对于小的帮助文件我一般是从 1 开始编号，递增直到帮助中的最后一个主题。大帮助文件比较复杂，可能会碰到重

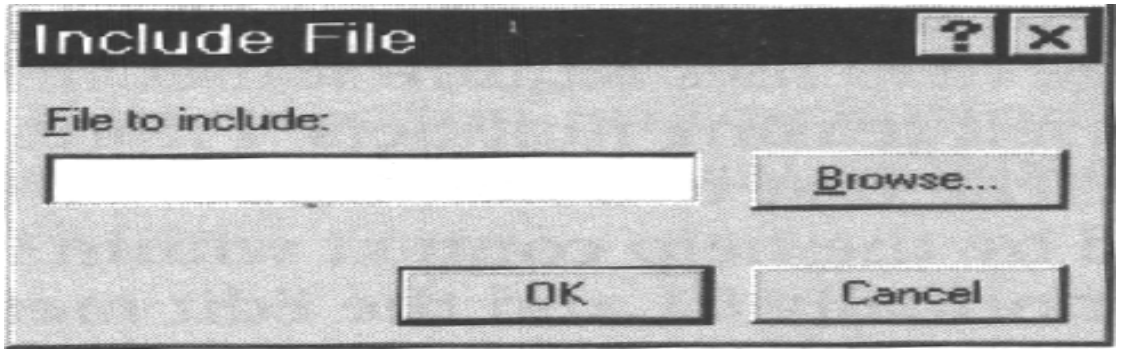
用序号的情况，这时，我一般用 3 或 4 位数的序号，前两位数是控件和菜单条在应用程序帮助内容中的位置。例如：菜单项 File 标为 01，菜单项 Edit 标为 02，File|New 命令的序号就是 0101，因为 New 命令项通常是 File 菜单中的第一项。第一个非应用程序主题赋值 0001，例如词汇表。应用程序窗体上控件的前两位数应大于最后一个菜单项的序号，我使用 Tab 序号作为控件标识的后二位数字，原因在于这类控件不像标签或其它的非活动组件，这些非活动组件不会出现在帮助文件中。

要增加新的映射入口页很容易，单击 Add，你会看到如下图所示的 Add Map Entry 对话框。

该对话框有三个域：主题标识符、被映射的数值（帮助内容中的序号）及注释。填完后单击 OK 就在工程文件中加入了一个新的映射。



正如前文所述，帮助文件中一般都包含 HM 文件，以减少工作量并提供快捷便利的检查方法。当帮助文件已测试并发行之后，才发现有一些很重要的主题没有覆盖到，再没有什么比这更难办的了。这时单击 Include 按钮，系统显示 Include File 对话框，如下图所示：



这个对话框提供了一个 Browse 按钮，可以用它调出已经保存在磁盘上的文件。单击后打开标准 Open 对话框，就像你在其它程序中用过的这种对话框一样操作。

编译帮助文件

完成工程文件后，就要进行编译了。如图 15.3 所示，单击主窗口底部的 Save and Compile 按钮。在编译过程中，Help Workshop 窗口会最小化，这样你就可以去做别的工作了（因为编译大的帮助文件需要很长时间）。编译完成后，会看到如图 15.4 所示的对话框：

你应该注意到这个对话框中立刻显示了一些东西——上图显示了帮助文件中存在一些错误（实际上那些都是注释，但绝大多数情况下还是应该把它们看

成是错误）。在帮助文件条目中，我们没有给出 HM 文件中的所有条目，即：在帮助文件覆盖过程中出现了漏洞。从中我们可以看到，使用 HM 文件很有帮助——至少会减少丢失帮助主题的机会。

那么 HM 文件就很完美了吗？远远没有。看一下表 15.2，那里曾经给 ID_FORMAT_FONT 增加了一个 Map 条目，这是 MFC 用在 AFXRES.H 文件中的标准标识符，不是包含定制控件标识符的 RESOURCE.H 文件，结果 HM 文件不能识别它——必须手工增加对标准控件的支持。

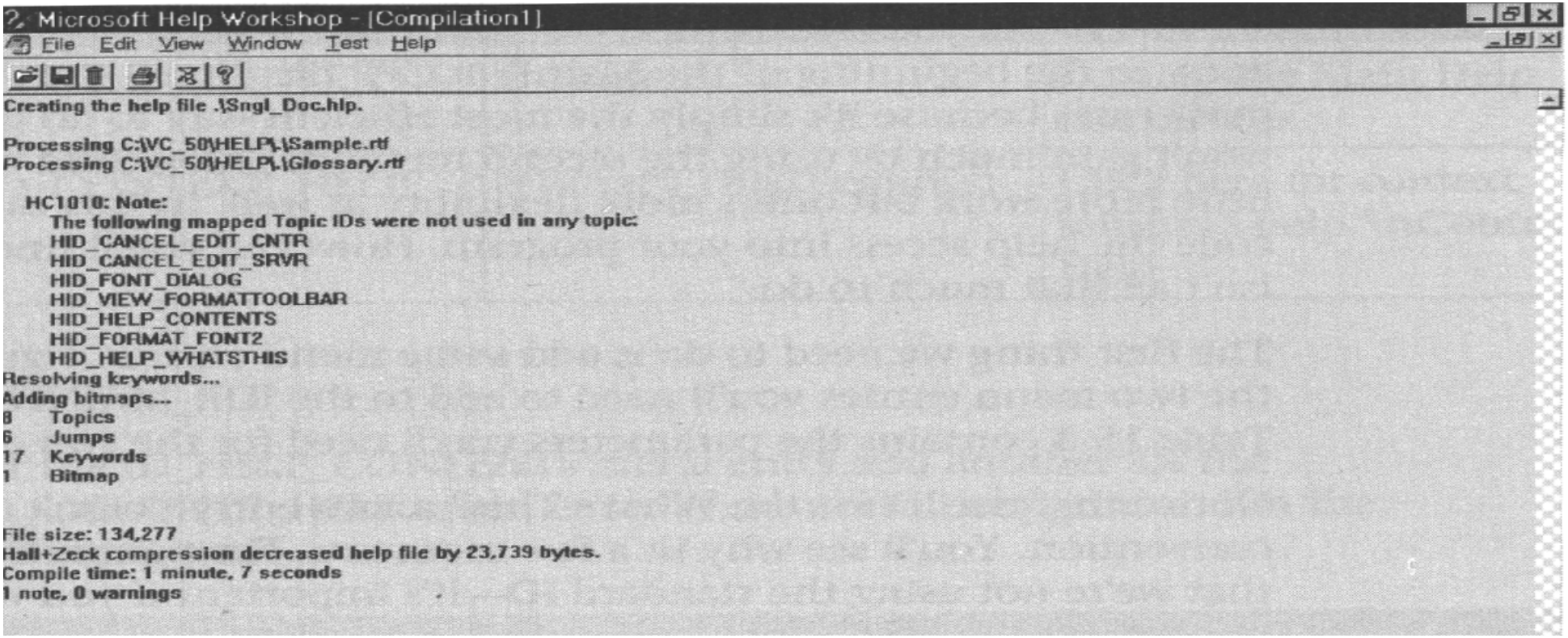


图 15.4　该编译结果对话框显示了帮助文件的当前状态及错误信息

15.5 为应用程序增加标准帮助

终于可以给我们的应用程序增加帮助了！这里选用第 2 章开始的单文档工程文件的资源版（第 3 章实现）作为例子，因为它非常完整，而且在以前我们没对它作任何帮助支持。必须认识到帮助支持的最重要部分在于帮助文件本身，我们要实现的帮助支持根本不需要多少编码。

实际上我们会看到两种实现方法，第一种方法是全自动的——只需在 MaiFrm.CPP 的开始处加入一个菜单项并修改适当的菜单项标识。这是最常用的方法，因为它最有效。第二种方法略微复杂些，但也更加灵活，需要在程序中手工加入访问帮助的代码，但实际上还是很简单的。

首先要增加一些菜单项。图 15.5 给出了要加入到 IDR_MAINFRAME 菜单中的两个菜单项，表 15.3 是要用到的参数。

表 15.3 帮助菜单参数

菜单标识（ID）	菜单标题	提示
ID_HELP_CONTENTS	&Contents\tF1	Display the Main Help File\nHelp
ID_CONTEXT_HELP	&What's This?\tShift-F1	Click here for context sensitive help.\nContext Help

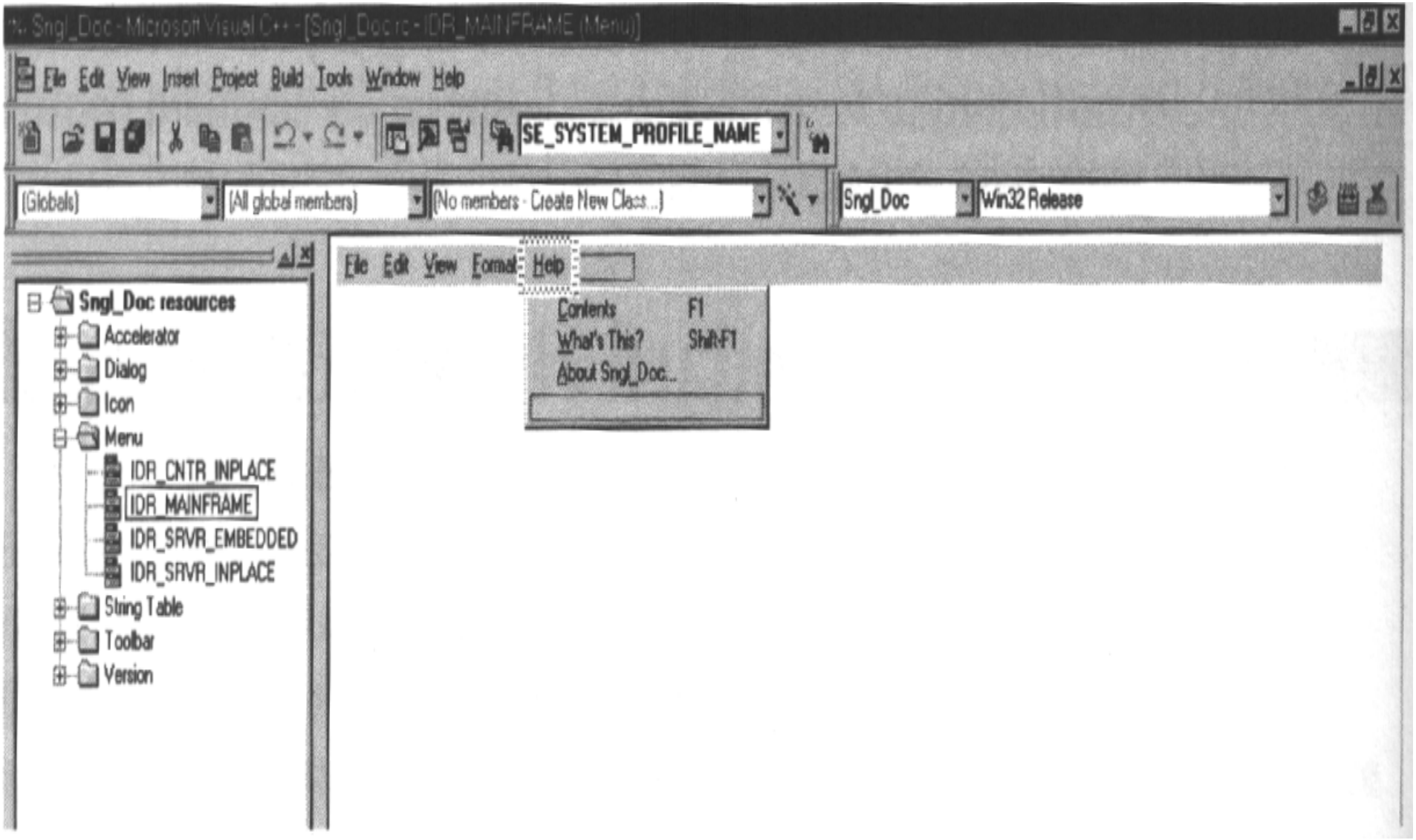


图 15.5　需要增加两个新的菜单项来测试本章前一节创建的帮助文件

注意 What's This?的 ID 号，这个菜单项没有使用通常的命名规则。一会儿就会讲到，现在，只需记住这一点——如果想自动实现 MFC 帮助，这个命名是非常重要的。

同时也要修改 IDR_MAINFRAME 工具栏上的 help 按钮，当前它显示的是 About 对话框，没有加入帮助支持时，这当然是很合适的；加入帮助时，打开 IDR_MAINFRAME 工具栏，双击 help 按钮，将 ID 号改为 ID_HELP_CONTENTS。

这两个菜单项当然也可以有与之相关的快捷键，所以还需要打开如图 15.6 所示的 IDR_MAINFRAME 快捷键定义窗口。这里我已经加亮了其中一个列表项，可以给 ID_HELP_CONTENTS 菜单项增加一个新的快捷键，给 ID_CONTEXT_HELP 菜单项增加一个快捷键，如下表所示。

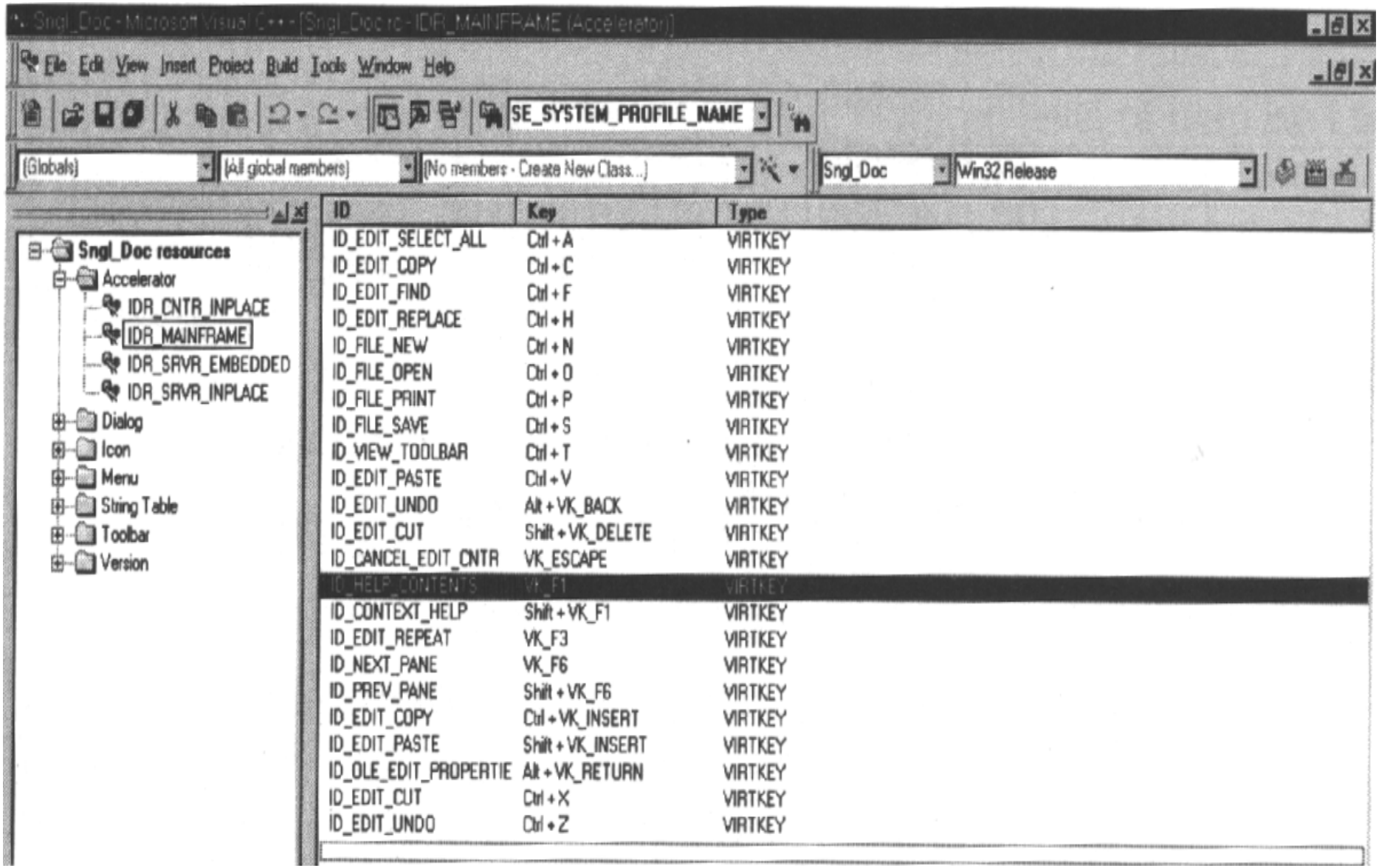
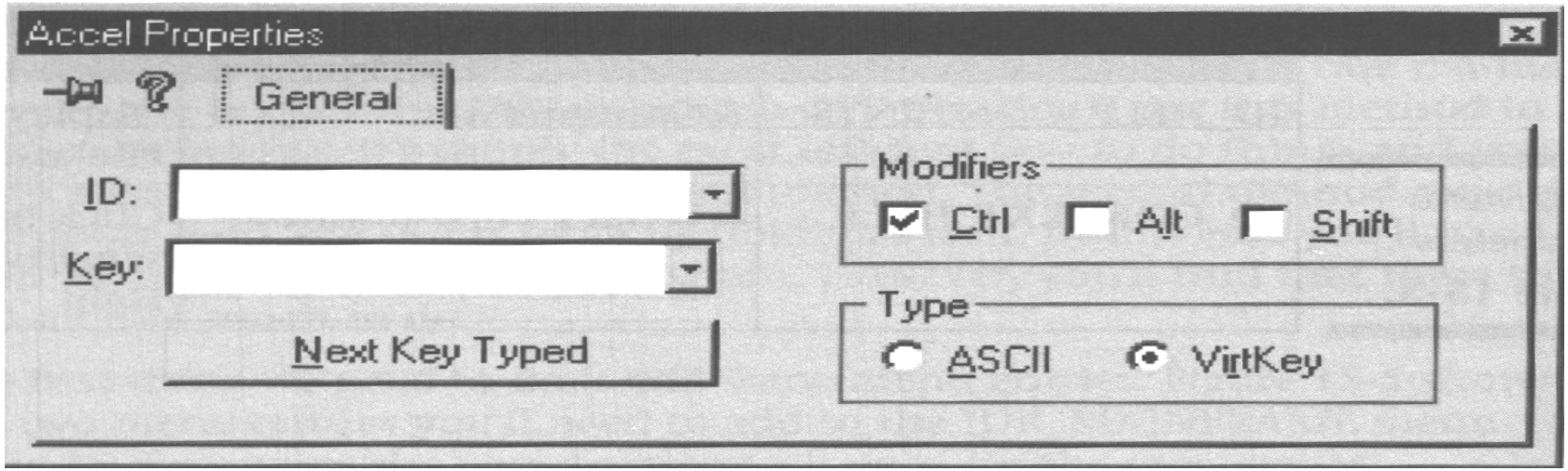


图 15.6　需要为这两个帮助菜单项定义快捷键，以便使它们对用户更友好

菜单标识（ID）	键	类型
D_HELP_CONTENTS	VK_F1	VIRTKEY
ID_CONTEXT_HELP	SHIFT+VK_F1	VIRTKEY

加入快捷键很容易，双击最后一个空白列表项处，显示如下图所示的 Accel Properties 对话框：



从下拉 ID 列表中选中所需的 ID 号，单击 Next Key Typed 按钮，按下分配给该 ID 的热键，这里 ID_HELP_CONTENTS 的热键为 F1，ID_CONTEXT_HELP 的热键为 SHIFT-F1。要确保选中了适宜的复选框，然后关闭该对话框。

记住我们说过，不用写实际的函数就能实现这些帮助选项。MFC 支持四个根本不用编程的缺省帮助动作：内容（Contents）、查找（Find）、索引（Index）和上下文相关（context-sensitive）功能。我们用内嵌的 MFC 函数实现上下文相

关功能。我们所要完成的全部工作就是加入程序列表 15.3 粗体部分所示的代码，该代码用于激活实现上下文相关功能的内嵌 MFC 函数。

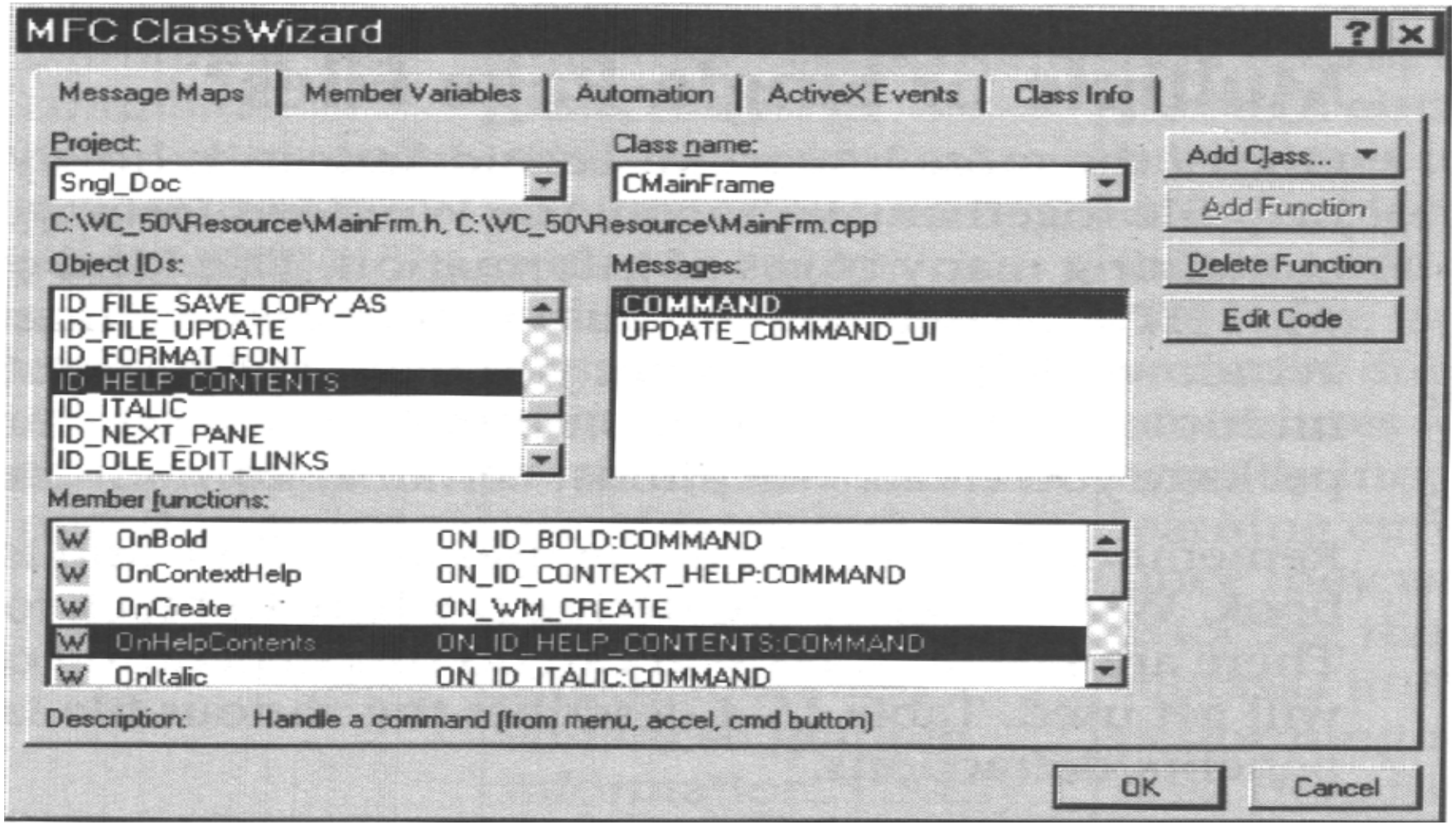
程序列表 15.3

```
BEGIN_MESSAGE_MAP(CMainFrame,CFrameWnd)
    //{{AFX_MSG_MAP(CMianFrame)
    ON_WM_CREATE()
    ON_COMMAND(ID_BOLD,OnBold)
    ON_COMMAND(ID_ITALIC,OnItalic)
    ON_COMMAND(ID_STRIKETHROUGH,OnStrikethrough)
    ON_COMMAND(ID_UNDERLINE,OnUnderline)
    ON_COMMAND(ID_VIEW_FORMATTOOLBAR,OnViewFormattoolbar)
    ON_COMMAND(ID_HELP_CONTENTS,OnHelpContents)
    ON_COMMAND(ID_CONTEXT_HELP,OnContextHelp)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

注　必须用选定的 ID 激活 MFC 帮助支持，如果在使帮助起作用时发生了问题，请检查菜单项的 ID 号。

现在要看一看 Contents 菜单项了。先要增加一个函数，用 View|ClassWizard 命令显示 MFC ClassWizard 对话框，选中 Message Maps tab 并选择 Class Name 域的 CmainFrame，在 Object IDs 程序列表中找到 ID_HELP_CONTEXT，在

Messages 程序列表中加亮 COMMAND 列表项，单击 Add Function，此时的 MFC ClassWizard 对话框如下图所示。



单击 Edit Code 按钮，就会看到函数框架，程序列表 15.4 显示了使 Contents 菜单项起作用所需的代码：

程序列表 15.4

```
void CMainFrame::OnHelpContents()
{
    //Call WinHelp to display the main help window.
    WinHelp (0,HELP_CONTENTS);
}
```

正如你所看到的，实现所需帮助的编码并没有什么特别之处，说实话，这段代码可能是本书中最短的代码。你所要完成的所有工作就是以预定义量 HELP_CONTENTS 调用 WinHelp()。这里我们不需要提供任何辅助信息，因此第一个参数被设置为 0。

15.6 建立基于 HTML 的帮助文件集

第八章已给出了一些建立基于 HTML 帮助文件必需的知识，HTML 帮助的基础元素是 Web 页面，就像在 Internet 上一样。然而，HTML 帮助又不仅仅是一个简单的 Web 页面，还要考虑许多其它因素。例如，要从访问的角度组织帮助文件。以下各节帮助你设计一个既有效又实用的帮助文件。

采用多页还是单页文件

注 基于 HTML 的帮助文件经常用多个文件存放帮助信息；这些文件的分布

形式很大程度上影响了用户查找时的便利性。

一个很有必要考虑的因素是如何组织所有帮助文件。使用 Windows Help 时，我们只建立了一个文件，它由许多信息页组成。这种方法不适用于基于 HTML 的帮助文件，因为此时不是用主文件的方法来实现的，总而言之，必须指出如何将多个信息页放入一个集合中，但所建立的集合必须包含所有的文件。

请谨记，帮助文件使用大纲方式组织，而且编排上与书籍的布局格式相似。你需要完成的工作是确定是从哪儿截断文件，完成这项任务的方法有多种，关键取决于应用程序中如何使用，表 15.4 描述了各种文件布局选项以及相应的优缺点。

表 15.4　HTML 文件的布局方法

布局方法	描述	好处	缺点
大菜单/对话框控件	对话框中的所有控件或树状菜单中的全部菜单项都放在一个文件中，使用锚地的方法可以让用户从一个主题切换到另一个主题，这种方法包含一个目录，让用户	初次使用这种布局方式的用户会觉得很容易浏览整个程序，而且，这种设计使帮助文件与使用手册很相像，目录的使用方便了信息的查找	过长的文件可能会使用户觉得下载开销过大而转去下载其它文件，而老用户又会觉得在一个文件中多次选定菜单项很麻烦，而且，由于修改时可能要对多个文件做相同的改动，因此会造成更新困难

	直接切换到所需主题		
大任务	每个文件只含有对完成某一特定任务的完整描述和过程。例如：对建立新文档的操作建立一个帮助文件	初次使用这种布局方式的用户会觉得很容易完成具体任务，而且每项任务都有独立的完整性，减少了程序变动时带来错误的机率；而且可以包含屏幕快照和图像信息；最后，这是唯一一种可以产生简单的教程帮助文件的编排方式，由于 Internet 带宽和灵活性的限制，降低了我们能提供的培训的能力	高级用户会觉得，经过许多程序项目才能查找到一条所需的信息很麻烦。缺少常规的子程序会增加用户的学习难度，因为他们不能把执行某一特定任务和所要调用的常规子任务联系起来。由于有冗余信息，帮助文件难于更新，并占用更多的服务器空间；增长的文件长度也使远程下载更加费时
小菜单/对话框控件	每条菜单项或对话框中的一个控件放在一个单独的文件中	较短的下载时间使得用户即使使用拨号方式上网也会愿意使用这种帮助。由于用户只看到一个屏幕会觉得查找信息很快。由于信息只出现	由于每条菜单项或对话框中的每个控件相应的信息都需要下载，初次使用的用户的积极性会大受打击，特别是用拨号上网时。目录不再是自动的，

		在一个文件中，更新快捷可靠。帮助文件在服务器上的总量减少了	需要单独的维护时间
小任务	建立任务清单和内容的树型结构表。每个任务被分成一系列易于实现的常规子任务，在任务总述中包括全部的特定事件	使用目录使得信息便于查找，特别是对于高级用户。树型结构形式使得用户可以直接找到所需信息。由于减少了下载时间，用户会更加愿意使用这种帮助。更新由修改每个子任务文件夹下的信息完成	可能会在任务处理过程带入微妙的错误，例如：菜单项可能在应用中的某一部分已经改变了，而另一部分则没变。一些新用户则会觉得面向任务的方式难于理解
主题型	每个文件包含一个帮助主题。例如：关于数据录入方法部分可以组成一个文件	想查找关于某个主题的用户会觉得十分方便，因为所有信息下载一次就完成了	对于信息组织方式依赖于菜单的远程用户，必须在所有菜单项信息下载完成后才能使用。因为一个菜单的变化需要查阅几个文件。信息更新也比较困难，原因在于一个菜单变化时就需要更新多个文件

注 即使是基于 HTML 的帮助文件在为用户提供信息时，也必须考虑到拨号连接存在的限制。

一个很有趣的现象是：使用基于 HTML 帮助文件的厂家总是觉得用户是用 T1 线路接入 Internet 的，实际情况是用户常常是用拨号方式下载文件。在第八章我们已经讨论了一些关于拨号连接的情形，最重要的是要限制每个文件的大小，这样用户才会愿意下载。

与浏览器不同，应用程序中不会含有 stop 按钮，这就意味着用户必须选择是等着每页从 Internet 服务器下载完再享用帮助，还是在帮助文件过长时拨打帮助热线。多数情况下我还是建议你组织好一个强大的技术支持小组，准备为那些用了长 HTML 帮助文件的应用程序提供支持。

注 在基于 HTML 的帮助文件中要慎用图像，因为这会极大地延长下载时间。

考虑 HTML 帮助文件的大小意味着：需要考虑是否使用响铃和报警来增强基于 HTML 的帮助文件的效果。例如：大部分 Windows 帮助文件使用全彩色的屏幕快照以帮助用户获得最易理解的帮助信息；另一方面，对于基于 HTML 的帮助文件就使用了黑白方式的图片以减少下载时间。

还需要控制 HTML 帮助文件中图像信息的使用数量。在颜色和/或分辨率上的减少毕竟是有限的，如果图像因此变得很不易识别，用户就很不值得下载了。换句话说，不论建立哪种形式的帮助都必须保证图像的质量。

结束这种无益争论的最好办法是先进行用户调研，一些用户可能只需要文字，可以在帮助屏幕上作一个选项供他们选用；另一些用户宁愿使用低质量的

黑白图片，还有一些用户则选择高质量的彩色图像，因此最好提供多种方式供用户选择。这时要考虑到用户的拨号速度，毕竟有些用户确实用 T1 连接方式，他们应该得到高质量的帮助文件。

查找能力

当用户经过了初级阶段之后，就很关心帮助文件提供的对所需信息的查找能力。他们要找的可能只是完成某项任务中的一小步，或是某个对话框中一个小控件的局部说明。糟糕的是，HTML 帮助文件不是全局可查找的，这个问题是所有供货商都必须面对的。

有些方法可以模拟 Windows Help 的查找方法，实现起来也不难。虽然不能完全实现 Windows Help 的查找能力，但可以尽量接近。下面是一些供参考的方法：

建立索引
定义附加的链接
增加目录
生成热门主题列表

注 基于 HTML 的帮助文件缺少按字查找的能力。

基于 HTML 的帮助文件不像 Windows help 那样，它缺少按字建索引的能力。这就意味着用户可能会提出许多预想不到的问题，而帮助文件的查找功能又无法问答。这时一般要提供某种形式的反馈机制，以便于用户提出其它链接的需求。另外，如果你的 HTML 帮助文件查找功能足够灵活的话，建立附加链接不

会是件频繁的事。

技巧 克服 HTML 帮助文件查找能力不足的一种方法是使用 Windows NT IIS version 4.0 及以上版本。它提供了对 Web 目录下所有文件建立索引的能力，在基于 HTML 的帮助文件上设置查找页面并不广受欢迎，特别是使用 Microsoft 提供的通用查找页面。

15.7 向应用程序增加基于 HTML 的帮助

向应用程序增加基于 HTML 帮助过程分几个步骤完成，前面几节已经讨论了第一步：考虑 HTML 文件的设计和布局。下一步是建立 HTML 文件本身，最后在应用程序中加进必要的链接。

在第八章中，我们已经讨论了建立 HTML 文件的基础知识，这里不再赘述。当然要想从脚本方便地转化为好的帮助文件，还不得不借助于一个适宜的编辑器，比如 FrontPage 或 Visual InterDev。这时，脚本用 Word for Windows 编写也无妨，否则需要附加的产品如 Map This 在图像和各种帮助主题间建立链接。这个过程相似于我们前面做过的用 Hotspot Editor 执行编辑的过程。

我们还用前面章节中用过的作 Window 帮助的例子为例，唯一的区别是用 Word for Windows 将它转换成了 HTML 格式。理论上说，可以使用任何工具建立帮助文件，但用 Word for Windows 建立的主文档既可以用于印刷出版物，又可以用于电子出版物。我们还用那节中同样的示例程序，但把原始的资源工程

文件拷贝到另一个目录以保持帮助例子的独立性（屏幕快照上的目录名可能与机器上实际的目录名不匹配）。

注释 可以在以下站点 http://www.osborne.com 找到本书中的所有工程文件、帮助文件的拷贝。它们会帮助你更好地准备帮助文件。

与本章中 Windows 帮助格式的示例不同，没有自动实现向应用程序中加入基于 HTML 的帮助文件的方法，只能为帮助文件中的每个访问点手工编写代码。我们的例子看上去很像本章第一节中的 Windows help 访问的第二种形式，而不是第二节讲述的方法。

这个示例由在帮助菜单中增加两个菜单项开始。精确的过程可以参见前面小节讲解的如何向应用程序加入标准帮助的方法来实现。菜单项的格式如图 15.4 所示，表 15.3 则给出了用到的参数。当定义了菜单项后，分别给 ID_HELP_CONTENTS 和 ID_CONTEXT_HELP 菜单项添加新的快捷键，如图 15.5 所示（前面在加入标准帮助时，我们已经讲述过了）。

注释 本章介绍了在应用程序中使用基于 HTML 的帮助文件的一种很传统的方法（不包括上下文相关帮助，但添加上下文相关帮助的方法与添加基于 HTML 的帮助的方法相似）。还有其它增加基于 HTML 帮助方法，在 12 章中我们会讲述，那时，你可以不用本例中所用的访问主页的方法而是在 Web 站点上设置一个帮助桌面。

首先，在应用程序中加入一个新的对话框。在 ResourceView 中右击 Dialog 文件夹，从上下文相关菜单中选择 Insert Dialog，这时会看到一个标准对话框，

右击该对话框，从上下文相关菜单中选择 Properties，系统显示 Dialog Properties 对话框，将 ID 域改为 IDD_HELP_DLG（或其它你喜欢的名称作为帮助对话框的名字），将 Caption 域改为 Main Help（或其它你喜欢的显示在对话框标题栏的文字）。本例中还需要改变对话框的某些其它属性。在 Styles 属性页选中 Minimize 和 Maximize 复选框，还需选中 Border 域的 Resizing 选项，使用户能够重新调整结果对话框为适宜的大小。关闭 Dialog Properties 对话框。

现在有了一个用于显示的新对话框，这时要加入一个 Web Browser 控件。使用 Project|Add to Project|Components and Controls 命令来显示 Components and Controls Gallery 对话框。双击 Registered ActiveX Controls 文件夹，加亮 Microsoft Web Browser 列表项，单击 Insert 向工程文件中加入这个控件，当提示是否确定时选择 OK 按钮。完成上述工作后，关闭该对话框。

本例要用到 5 个命令按钮和一个 Web Browser 控件。有时一图达千言，所以用图 15.7 来说明我对帮助对话框所做的配置。这是使用基于 HTML 帮助时你应该向用户提供的最小必要配置，原因在于用户总是需要在各个帮助屏幕来回翻页的。表 15.5 给出了该对话框中所有控件的属性列表。

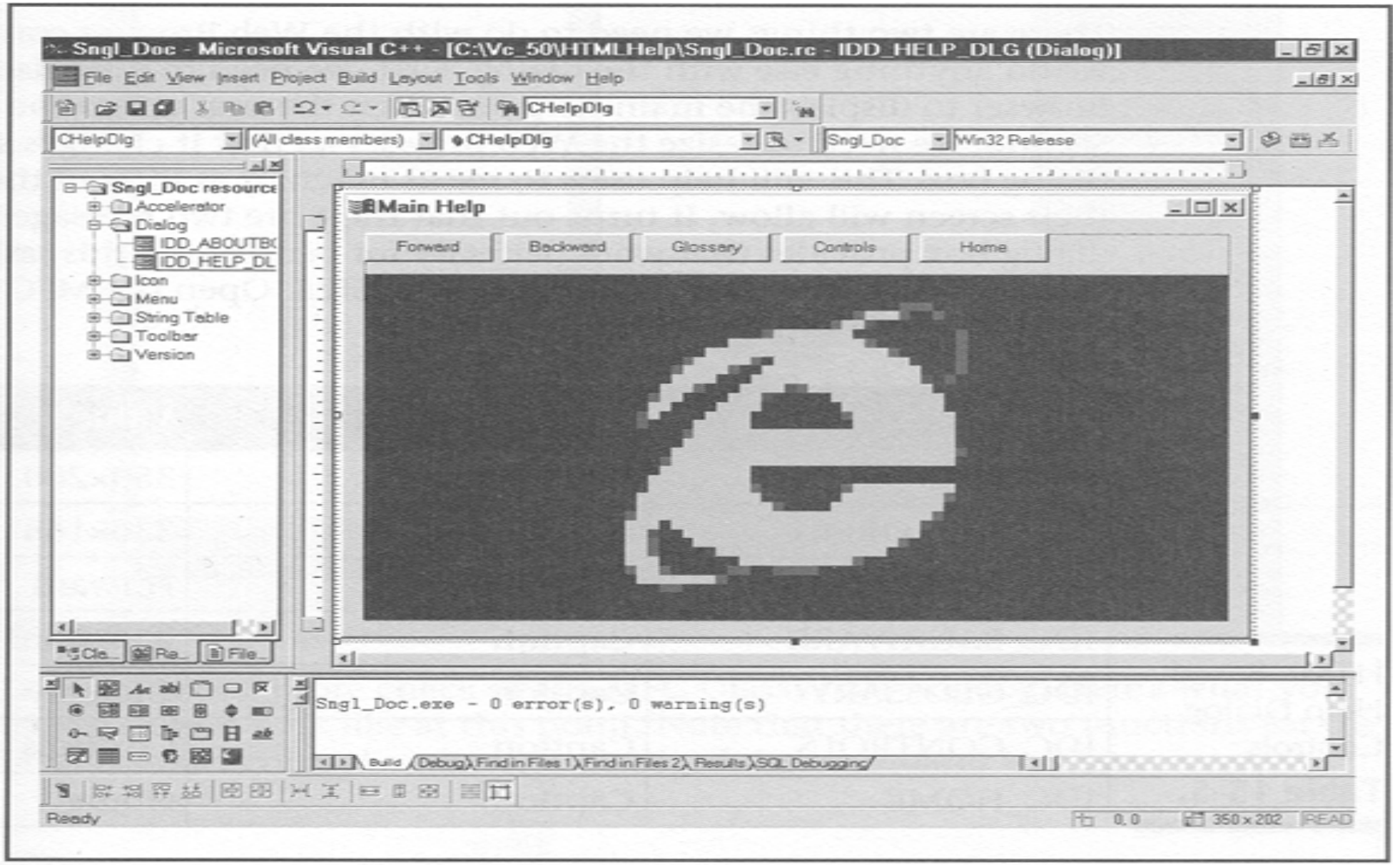


图 15.7　一定要把帮助对话框设计得易于使用

表 15.5 基于 HTML 的帮助对话框控件

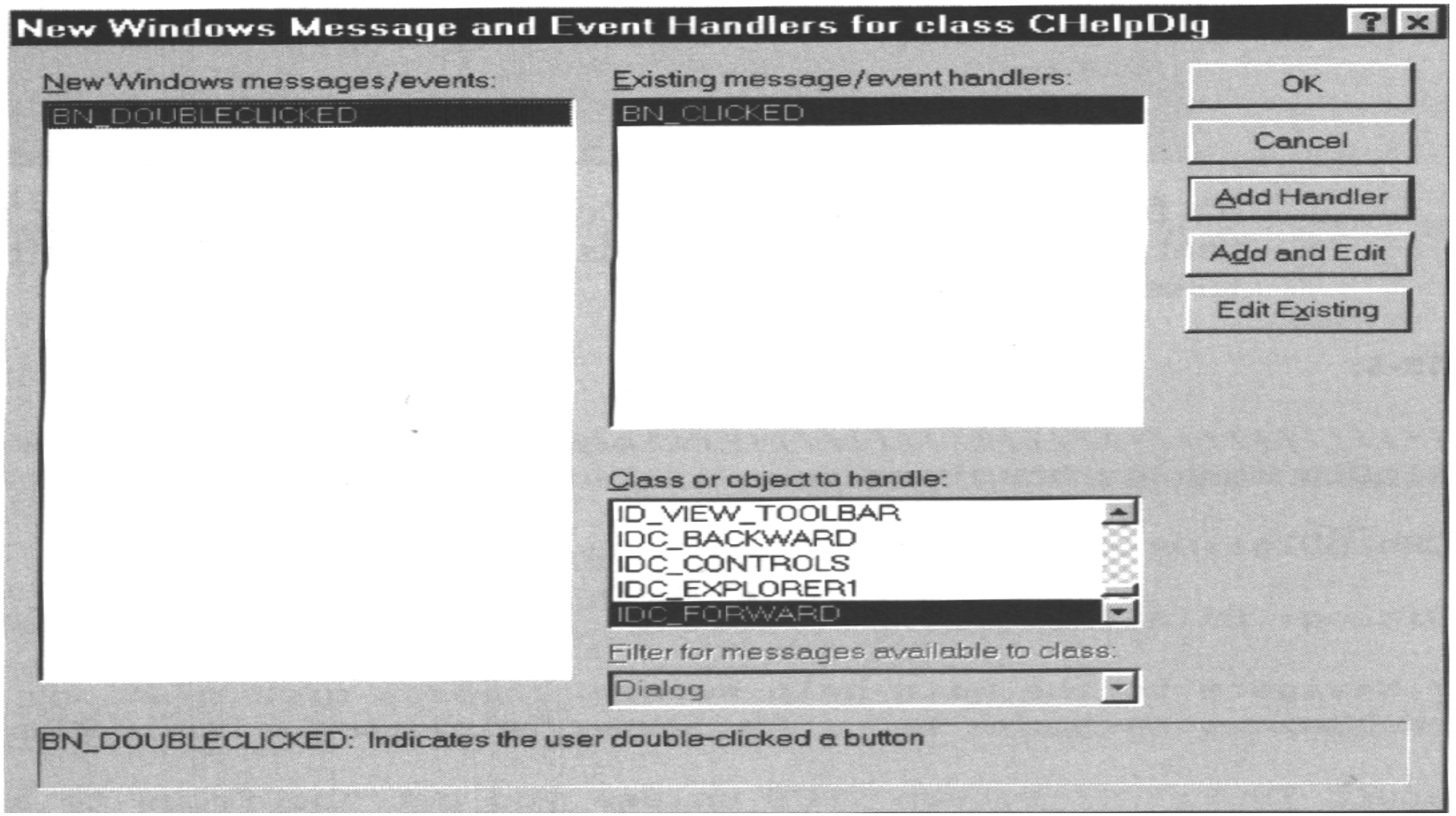
控件	属性	取值
IDD_HELP_DLG	Size	350x200
IDC_EXPLORER1	Size	336x168
IDC_FORWARD	Caption	Forward
IDC_BACKWARD	Caption	Backward
IDC_GLOSSARY	Caption	Glossary
IDC_CONTROLS	Caption	Controls
IDC_HOME	Caption	Home

建立一个对话框并不代表在应用程序的各个部分都可以访问它，需要为它建立一个类，这也是我们将要完成的工作。按住 CTRL 键，然后双击该对话框（同时保证没有双击其中的任何控件），系统显示 Adding a Class 对话框，单击 OK 建立新类，输入 CHelpDlg 作为类名，并选择 CDialog 作为基类，按 OK 创建新类。关闭 MFC ClassWizard 对话框（在完成下面的几个安装步骤后再加入所需的代码）。

此时，虽然该对话框已经可以访问了，但对话框上的控件还都不能访问。我们需要处理的第一个控件是 Web Browser。双击 IDC_EXPLORER1 控件，系统显示 Add Member Variable 对话框。在 Member Variable Name 域中键入 m_WebBrowser，在 Category 域中选择 Control，在 Variable Type 域中选择 CwebBrowser2，单击 OK 完成变量创建工作。做上述工作的目的是为了在应用程序中使用 Web Browser 控件。

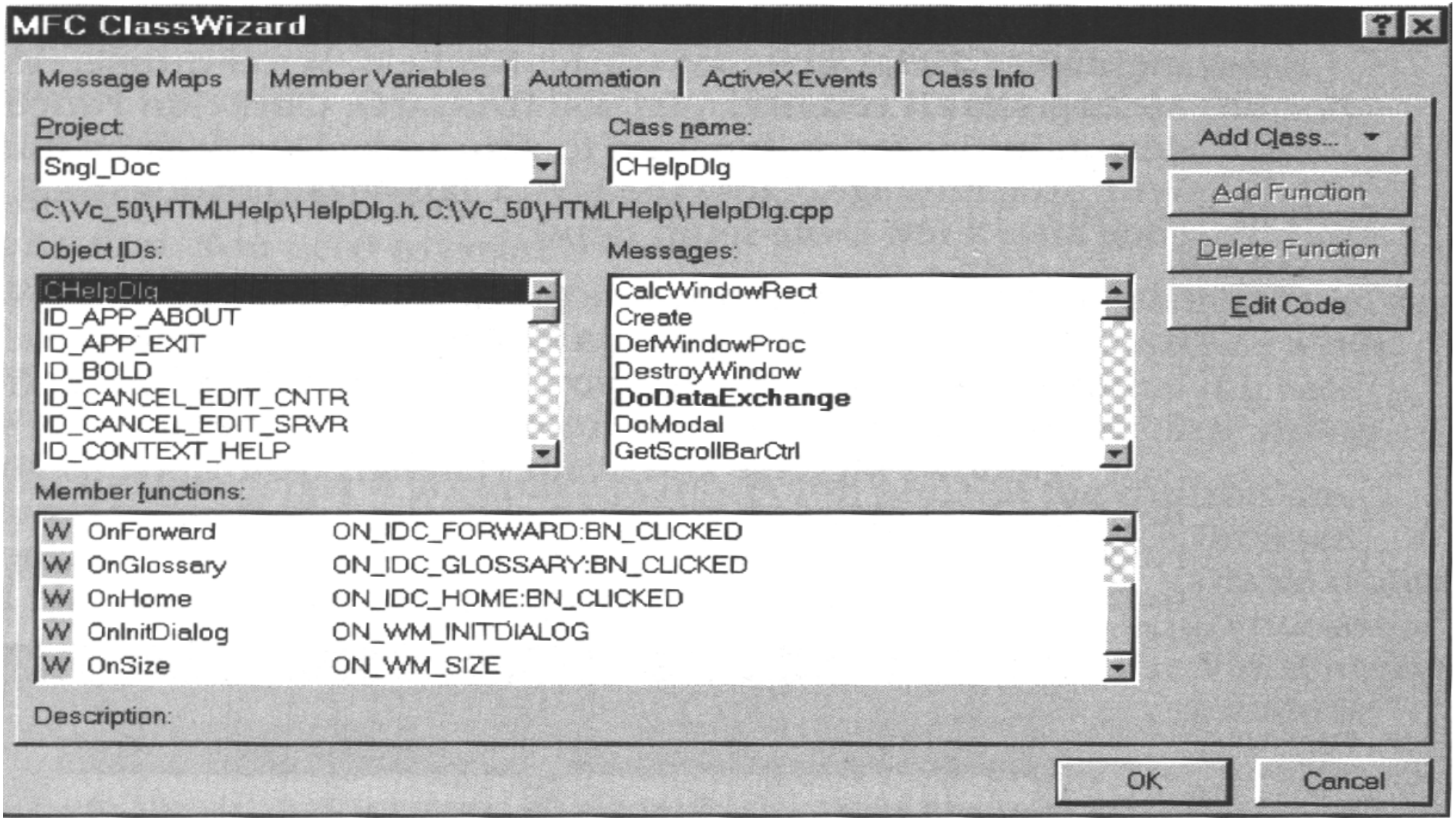
在处理该对话框的其它任务前，对 Web Browser 控件我们还要完成两项任务。其一是，我们需要初始化 Web Browser 控件，以便在用户打开该对话框时它显示主帮助文件。其二是，我们需要改变 Web Browser 控件的大小，以便它与对话框的大小一致。这样用户就能够根据屏幕大小和帮助文件的大小看到尽可能多的帮助内容。完成这项任务只需要使用对话框提供的两个事件，一个是 WM_INIDIALOG，另一个是 WM_SIZE。使用 View ClassWizard 命令打开 MFC ClassWizard，选择 Message Map 选项卡，然后加亮 Object IDs 域中的 CHelpDlg，滚动 Message 列表，直到找到 WM_INIDIALOG，单击 Add Function 在你的代码中添加缺省的初始化函数。对 WM_SIZE 消息完成与 WM_INIDIALOG 消息相同的工作。关闭 MFC ClassWizard 对话框（在完成了随后的几个设置步骤后，我们将添加所需的代码）。

用户当然需要五个按钮执行一定的任务，实现起来相对比较简单：右击第一个按钮（也就是 Forward 按钮），从上下文相关菜单中选择 Events，你会看到 New Windows Message and Event Handlers 对话框，加亮 BN_CLICKED 消息，然后单击 Add Handler 按钮。关闭 New Windows Message and Event Handlers 对话框。对所有五个按钮重复上述操作步骤，下图是 New Windows Message and Event Handlers 对话框的外观：



让我们再看一看 MFC ClassWizard 对话框，下图是当前状态下该对话框的显示信息。请注意，Web Browser 控件有两个函数，每个命令按钮都有一个函

数 。



现在到了给我们的示例工程中添加代码的时候了。程序列表 15.5 显示了示

例程序中需要添加代码，在你看过之后我会再进行讲解。

程序列表 15.5

```
////////////////////////////////////////////////////////////////////////////////////////////////
//CHelpDlg message handlers
BOOL  CHelpDlg ::OnInitDialog()
{
     CDialog :: OnInitDialog();
     //Navigate to the main help screen.
     m_WebBrowser.Navigate("http://nt_server/help2/sample.htm",          NULL,NULL,
NULL,NULL);
     return TRUE;      // return TRUE unless you set the focus to a control
                       // EXCEPTION:OCX Property Pages should return FALSE
}

void  CHelpDlg ::OnSize(UINT  nType,  int  cx,  int  cy  )
{
     CDialog :: OnSize( nType,  cx,  cy  );
     // Change the WebBrowser control size to match the dialog.
     m_WebBrowser.SetHeight( cy - 64 );
     m_WebBrowser.SetWidth( cx - 28 );
}
```

```
void   CHelpDlg ::OnForward()
{
     // Go to the next site in the history list.
     m_WebBrowser.GoForward();
}

void   CHelpDlg ::OnBackward()
{
     // Go to the previous site in the history list.
     m_WebBrowser.GoBack();
}

void   CHelpDlg ::OnGlossary()
{
     // Go to the Glossary Web page.
     m_WebBrowser.Navigate("http://nt_server/help2/glossary.htm", NULL,
                                   NULL, NULL,NULL);
}

void   CHelpDlg ::OnControls()
```

```
{
    // Go to the Controls (Format) Web page.
    m_WebBrowser.Navigate("http://nt_server/help2/format.htm", NULL,
                                    NULL, NULL,NULL);
}

void  CHelpDlg ::OnHome()
{
    //  Navigate to the main help screen.
    m_WebBrowser.Navigate("http://nt_server/help2/sample.htm", NULL,
                                    NULL, NULL,NULL);
}
```

正如你所看到的，这些代码丝毫没有神秘之处，特别是当你阅读过 12 章的例子之后更是这样。所有的函数都与所建立的 Web Browser 内存变量（m_WebBrowser）打交道。在 OnInitDialog()函数和 OnHome()函数中都使用了 Navigate()函数，使浏览器显示主帮助文件（SAMPLE.HTM）；类似地，OnClossary()函数和 OnControls()函数使用 Navigate()函数来显示指定的页面；OnForward()函数和 OnBack()函数则顾名思义，它们调用了 GoForward()和 GoBack()使得用户在 Web 页面间切换。最后一个函数是 OnSize()，它用 Web Browser 控件的 SetHeight 和 SetWidth 属性来匹配对话框的大小。当用户改变对话框大小时，Web Browser 控件的大小也随之而变。

这样，我们就有了一个功能完善的帮助对话框了，但还是无法访问它。将它加入应用程序是很简单的，首先，像下面的黑体部分那样在 MainFrm.CPP 文件中加入帮助对话框：

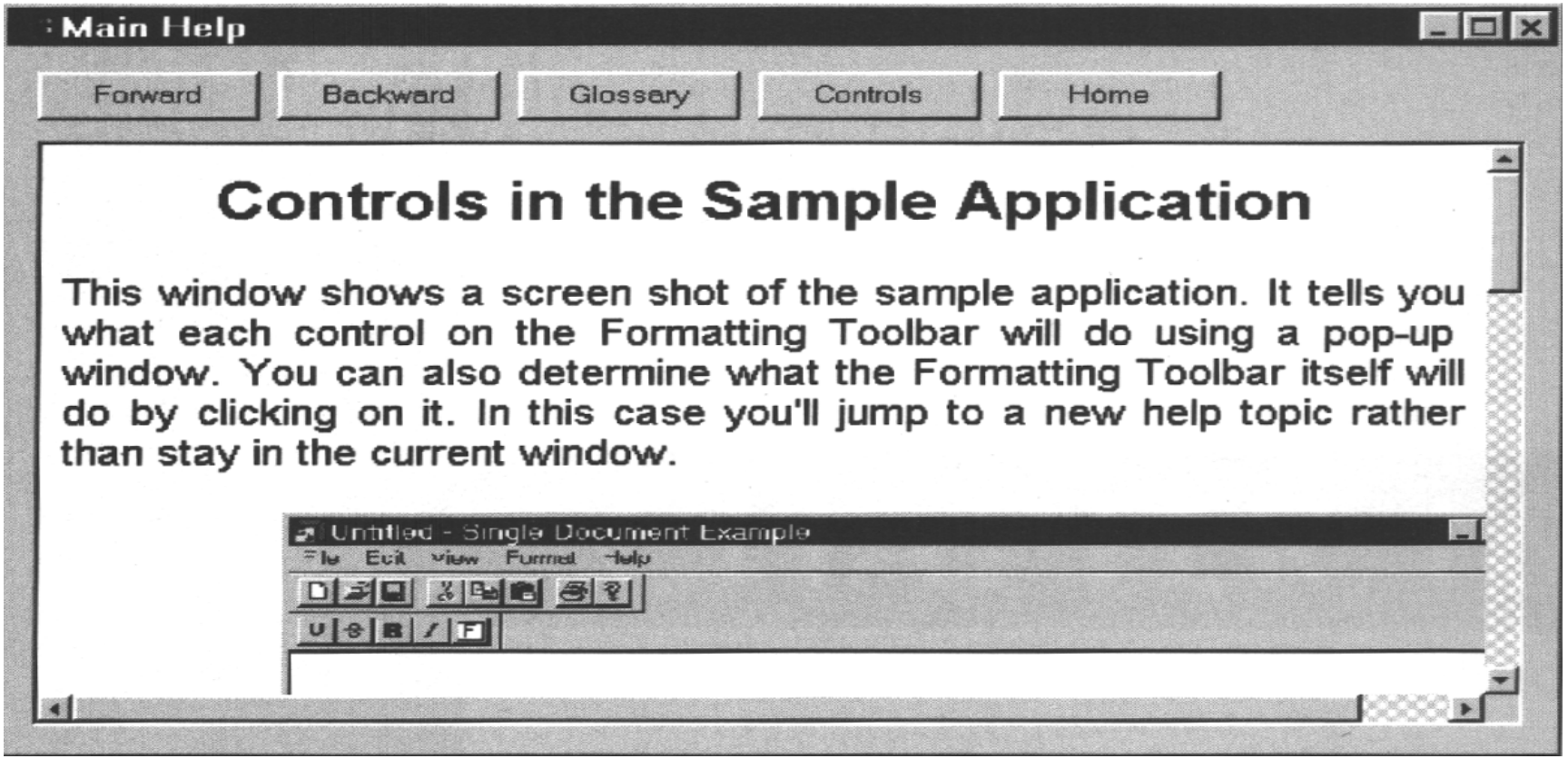
```
//  MainFrm.cpp  :  implementation of the CMainFrame class
//
#include  "stdafx.h"
#include  "Sngl_Doc.h"

#include  "MainFrm.h"

// Include our help dialog.
#include  "HelpDlg.h"

#ifdef  _DEBUG
#define  new  DEBUG_NEW
#undef  THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
```

我们还需要在 OnHelpContents()函数中添加一些代码，具体代码为：

```
void CMainFram :: OnHelpContents()
```

```
{
    CHelpDlg    OHelpDlg;    //Create a help dialog instance.
    //    Display our help dialog.
    oHelpDlg.DoModal();
}
```

如果此时编译并运行这个应用程序，你就可以使用 Help|Contents 命令访问基于 HTML 的帮助。下图是帮助对话框的一个示例：



第 16 章　打包应用程序

“打包应用程序”，听起来就像是打包物品送到商店中一样。从程序员的角度来看，打包与商店无关，但对用户使用时的感觉却至关重要。无需讳言，商店中最吸引我们注意力的是商品的包装，只有当打开包装之后，用户才会关心其中的内容。如果用户在安装应用程序时，就发现想要的东西都具备了，就会很愿意接下去继续使用——好的开始是成功的一半。

打包对于用户能够快速地安装程序、轻松地使用程序也很重要。设想一下一辆没有仪表盘的小矫车，它肯定能够前后左右地调整方向，但没有仪表板指示灯告诉司机正在走哪个方向；理由是设计师当然知道运动的方向，还安装仪表板灯干嘛!? 这样的汽车肯定卖不出去！同样道理，一个没有好的帮助的应用程序，看上去像是让用户猜测能用它来干些什么——因为如果只是简单地加入了一个 README 之类的文件，用户是根本不会去看的。

注释　第 15 章讲述了建立良好的帮助文件的必要性。如果你还没有给你的应用程序作一个好的帮助文件，那么在阅读本章之前你应该首先创建一个帮助文件。我是好的帮助文件的积极拥护者——它们会节约你自己和用户大量的时间和精力，本章的前提是假定你也是好的帮助文件的拥护者。

注 要保证应用程序界面的实用性和一致性（包括安装程序）。

有些程序员只是简单地将所有程序文件放在一个软盘上，然后作一个批处理文件，就万事大吉、开始发行了；更有甚者，如果该程序只是内部应用，甚至连测试都不测试。过不了多久，他肯定会接到许多不满的用户电话，软盘上可能少了某个重要的文件，用户执行批处理后根本运行不了。没有人会再喜欢这种程序——即使是重新加入了一个好看的界面、安装程序并修正了错误。其实并不是程序本身有什么问题，而是程序设计者的态度不够认真。第一印象对一个程序的成功是很关键的！

不管谁用你的程序，都要在程序出门之前完成正确的包装。少做一点都会成为困扰你及用户的隐患。没有人愿意买一个不完整的小矫车——同样的道理，也没有人想要一个不完整的程序。花点时间做一个好的打包吧！——即使程序是给其它的程序员或高级用户使用的。人人都会受益于一个功能完善的程序！

如果我没有花点时间讲述程序的打包就是我的失职，这就是本章的目的。在我看来，导致一个应用程序给用户留下深刻印象的最重要的两个因素是：理解各种打包类型，为相应的打包建立安装程序。我还会讲到不同环境下程序员应特殊考虑的因素，例如，内部程序极少考虑开发共享软件的人所要考虑的共享问题。

Web 链接 本章我们使用了 InstallShield 6（Visual C++ 6.0 提供的 Free Edition）。可以在 http://www.Installshield.com/ 站点找到 InstallShield 6 和许多有用的用于打包的产品。不幸的是，至少在本书写作时，该软件还不能免费下载，然而，可拨打(800)374-4353

或 (847)240-9111 以了解该 Free Edition 并获得其拷贝。

16.1 理解各种打包类型

在打包交付应用程序时，需要考虑许多事情。对绝大多数程序员来说最重要的是运行环境。内部使用的程序可能与通用程序考虑的侧重点不同，它可能没有时间完成一个完整的打包工作——对内部应用来说时间往往是至关重要的因素。

注 有三种基本的程序打包类型：企业内部应用软件、共享软件和商业软件。

本节将讲述三种最基本的程序打包类型：企业内部应用软件、共享软件和商业软件。你很可能只会去阅读其中之一，但最好还是也读一下其它类型的叙述，因为其中包含有许多有用的技巧和提示。而某些人更不应该满足于只了解某一种类型，例如：你是公司的顾问，你也就是半个内部程序设计人，因为所建立的大部分应用程序都是供公司内部使用的；同时，为了满足外部用户的需求，至少要提供一个共享软件级的打包封装（一些客户甚至想要一个商业软件级的打包，但这种情况相对少一些）。

提供哪种打包方式很难一言以蔽之，但也不是毫无依据，关键在于对应用程序的使用人群作出正确的评估。如果用户需要的只是运行稳定、不需要多少的修饰，使用企业内部应用打包格式就很合适了。

企业内部应用软件

设计企业内部应用程序的程序员最不用关心封装的华丽性，最需要关心程序的可定制性。事实上，企业内部程序不用靠精彩的画面、美妙的音响来吸引用户，界面只要功能完善、用户友好就足够了。如果你手边还有许多项目有待完成，即使费尽心机地设计了一个精美的安装程序，也不会赢得老板的赞赏。

另一方面，企业内部程序却常要考虑提供可定制性。对于可定制的程序，特别是数据库应用程序，特别之处并不只是要使用特定的文件，还需要定制注册项及进行许多其它的设置。建立一个在企业内部各种工作站上都运行良好、不需要设计人员帮助的安装程序是最重要的要求，必须花些时间手工调整用诸如 InstallShield 一类产品建立的标准安装程序。

企业内部安装程序不同于其它两类安装程序的另一个重要之处是发布时所使用的介质。共享软件和商业软件，通常使用软盘存放（现在越来越少了）。除非你所在的企业太落伍了，否则肯定会有 LAN 环境供发布程序，可以充分利用这个环境。同时使用最简单而有效的存储介质：CD 盘。你会发现如果将应用程序打包在 CD 上，用 LAN 来发布，建立和测试安装程序都会快得多。

技巧 InstallShield 6 Professional Edition 提供了通过 Intranet 安装应用程序的能力。不管公司的规模有多少个楼宇，都可以应用我们介绍的原理，通过 LAN 发布应用程序。在决定以软盘或 CD 的形式存储应用程序并手工发布之前，必须花时间研究 InstallShield 或其它封装程序提供的各种发布方法。对于企业内部应用程序来说，用 LAN 或 intranet 来发布最为有效。

在 LAN 上打包好应用程序之后，只需在登录服务器的主批处理文件中加入安装程序命令就行了。下次当用户登录时，批处理检查该应用是否已经安装；没安装，自动调用已放在 LAN 上的安装程序。

技巧 安装公司内部软件时，用 UNC 路径代替标准驱动器符的方法非常方便。源目录用 UNC 路径保证每个用户都能方便地访问所需的源文件。

在企业应用环境中，发布程序时还需要考虑一些因素，下面各要点就是为保证安装程序的可定制性所要考虑的。使用哪种方式取决于具体环境，如企业机构的设置以及用户的连网情况。

通用文件可以集中存放 应用程序的标准发布方式为将所有文件都安装到用户的机器上，这是共享软件和商业软件发布的唯一方法；而对于企业内部程序来说，可以把通用文件——比如 DLL——存放到文件服务器上。这种方法有两个好处：首先减少了对用户机器存储空间的要求；其次当用户在 DLL 中发现错误时，更改起来更快——只需改动服务器上的 DLL 就行了。这种方法也有两个缺点：第一，用户必须连网；第二，如果经常加载/卸载 DLL 必然会极大地增加网络流量。

绝对的优势 共享软件和商业软件的设计者无法预先假定应用环境的任何因素——源和目的都不确定：使用何种机器是个未知数，有时连操作系统也是未知的；而对企业内部应用软件来说，设计者则没有这种麻烦，所要顾虑的是网上机型互不相同。如果对所有机器都用同样的安装方式，可以指定一个缺省的目的目录；使用 UNC 路径名时，可以指定一个绝对的源目录，因为服务器路径名不会因安装次数的不同而变化。当

网络规模不大、工作站配置一致时，甚至可以规范到每台站点。总之，不需要给用户提供太多的安装选项，从而从根本上减低安装程序的复杂性。程序员所要完成的工作就是事先胸中有数。

不用设立配置选择项 所有的商业软件都提供三种安装时配置选项：定制安装（Custom）、典型安装（Typical）和最小安装（Compact）。定制安装允许用户选择程序部件；典型安装专为桌面用户设计；最小安装则是为膝上电脑用户设计的。多数情况下可减化为两项：膝上安装和桌面安装。其实，用这些词汇只会使用户费解最小安装的含义。

共享软件

共享软件的设计者在打包程序时所遇到的挑战最大。看看其中的一个主要问题：安装的大小。与拥有高速网络环境的企业内部软件不同，也不像商业软件那样可以用 CD 进行发布，共享软件的用户一般用低速 Modem 连接到 BBS、联机服务或 Internet 上。有足够的证据表明，当程序大小超过 1MB 时下载它的人数会急剧下降。

注释 本节所限定的文件大小只是指导性的，潜在用户愿意在你的程序上投入的盘空间和时间取决于许多因素，诸如人为的估计和程序已显示的价值等等。例如：有一个我很喜欢的图像打包共享软件，它要占 7MB 硬盘空间，压缩后也超过 1MB，但人们还是很愿意下载并使用它，因为它物有所值！众口皆碑就是人们愿意下载的最好动力。你需要考虑

的是人们是否愿意为你的程序掏腰包——如果你真想参与到共享软件市场的话，这是必须学习的经验。

好了，这些决定了程序空间的限制。那么如何克服呢？关键在于如何将产品市场化。多数成功的共享软件一遍又一遍地使用同样的图像和声音；换句话说，不会出现安装程序用了一种奇异的图像，而应用程序本身又用了另一种图像，共享软件的要旨是安装和使用的界面的一致性。事实上，使用精确的程序设计技术能使你有多次机会轻松地摆脱应用程序的图标（我们在第三章讨论过）。

像企业内部软件那样没有动人的图像或声音，共享软件也能赢得用户吗？情况不太一样。如果真是想吸引用户进一步购买，就必须在程序中加入一些靓点。没有人愿意付钱给一个看上去邋里邋遢的程序，即使它的功能很强大。显然，想让程序既短小又完美是件很难的事。

开发共享软件的另一个问题在资金方面。许多共享软件的作者只有一两个人，在他们等待产品售出的过程中不得不从事其它咨询或冒险性的工作来维持生计，他们没有充裕的时间，也不会像出品商业软件的大公司那样，有专业的美术或音响指导。因此，共享软件只要设计良好、占用空间合适（比如说 5MB 左右）并稍加雕琢就很令人满意了。

我们尚未讨论共享软件设计者必须面对的最大的问题，安装程序只是第一步。企业内部软件对应用环境有最全面的控制，可以控制到使用该程序的每台机器，而且设计者即使在机型检查和配置选项方面作的工作很少也会得到认可。商业软件的设计者情况相近，至少他们可以在软件包装盒上印上说明，规定运

行该程序的硬件要求，事实上，只要用户买了相应的商业软件就会意识到对硬件环境的需求。共享软件的设计者就没有这种保证了，它的用户的机型可是包罗万象的——甚至是一台老掉牙的 8088 PC。

这种无法预测会带来什么后果呢？首先，在程序中必须额外设计一段检测例程，以保证满足最小硬件需求。用户总是很少去读 README 文件（通常在这里指出最小环境需求），甚至根本不注意它，当他们安装失败时，却只会抱怨设计者，根本不会承认是自己的失误。因此，必须有检测程序。还需要提供灵活的、可供选择的配置选项，例如：当机器档次低时，用户可以只选用文本格式，摒弃所有的图像和声音。

注 企业内部的程序员能够全面地控制应用程序的运行环境，而共享软件的开发者则几乎不能掌握其程序的运行环境。

技巧 如果时间和财力充裕，设计一个卓越的安装程序，非常有助于共享软件的销售。要让用户明白钱花得物有所值，换句话说，如果在安装过程就吸引了用户，无疑增加了他愿意购买的欲望。

共享软件设计者一般都有一些使安装程序更加实用的技巧，下面我给出的一些原则也还是有帮助的。在最后决定使用什么样的包装之前，必须进行反复的试验比较。

打包分割化 减少盘空间占用量和下载时间的一种方法是将程序打包成几个部分。例如：主程序放在一个部分，图像在另一部分，声音在第三部分。这样用户就可以选择愿意付出的盘空间和下载时间的多少。必须保证程序没有图像和声音（或是其它决定放在单独包装中的部分）也可

以正常运行。安装程序也要提供相应的灵活性，当用户只下载了部分打包程序时应该也能安装。

帮助详细化 我们讨论一下将第 15 章建立的帮助文件放置在独立打包中的情况。方法之一是制作一个主帮助文件和一个详细的帮助文件。主帮助文件中含有对基本命令的解释，详细帮助文件中含有用户教程、宏语言描述和指令的细节说明。同样地，用户可以选择付出的盘空间和下载时间量。程序设计者也会从中受益：共享软件无法预测是否畅销，因此在没有回报时就投入大量的时间和金钱是很困难的，而用这种方法可以减少投资，直到有购买态势之后再继续投入。

技巧 已经有一些共享软件公司用模块化的方法拓展销售。例如：当用户下载时，ButtonWare 总是提供一个简单的帮助文件和程序的主要特征，购买时再允许用户下载全功能的程序和完整的程序文档。许多用户会抱怨称之为“半残的软件”，在购买之前根本无法测试！不幸的是，除非设计者能充分地说服用户购买，否则用户是轻易不会掏钱的（我说的是实话，无数共享软件公司都因为销售不畅而倒闭了，但它们的软件却仍在广泛使用）。

商业软件

我并不打算假惺惺地要教微软之类的大公司如何包装产品、开发市场——它们的市场开拓效果已经是有目共睹，我似乎在班门弄斧。商业软件区别于共

享软件的一个特点是软件的大小。大的软件公司通常集中于一个或多个产品，有许多专业的设计小组完成整合封装。研究一下这些软件，也会使我们这些普通的程序员从中受益。

过去的几个月中，每当我安装一个成熟的软件产品时我都非常留意。装过 30 个产品之后，我审视了一下这些记录，结果令人吃惊。例如：前面我说过，对于共享软件的作者，设计一个好的安装程序会有助于销售，因为他财力有限，必须用第一时间抓住用户。令人惊奇的是商业软件也情同此理，只不过方法不同罢了。用户已经购买了这种产品正在安装，但是还有其它增值产品呢？商业软件厂家总是利用安装的时机，推销相关连的增值产品（比如安装字处理程序时，介绍字典软件）。

另外，我发现商业软件在安装时总是告诉用户程序的新增特性。这倒不奇怪，因为这些公司早就注意到用户不留意 README 文件，安装往往就是用户装载一个软件的最后一步。同样地，他们要吸引用户，因此在安装程序中就要使用户对程序有一个整体概念。

商业软件总是带有许多多媒体功能也就不足为奇了，因为这些公司有足够的开发实力；即使不像这些大公司那样实力强劲，同样应该注意我们强调的该作什么，不该作什么。在安装程序中提供的版本虽然小但要代表原貌才能吸引用户，牢记你要作的就是尽可能给用户最好的第一印象，因此一个精彩的安装程序不仅仅是应用程序的良好开端，也是用户了解程序的窗口，它很可能会决定用户是否来购买你的程序。

你也能够从商业程序中吸取一些教训。其中之一是安装程序太复杂，以至

于测试得不完善，最近，我就遇到了这样一个产品，安装时发现它的帮助屏幕是给老版本做的——厂家忘了更新它了！结果呢？因为该产品也没带任何印刷的说明书，我就很困惑产品中的各部件有什么特性了。如果厂家认真完整地测试了安装程序，我就会明确地得到想要的信息，以决定安装哪些部件。

另一个很奇怪的问题是产品根本安装不上。似乎是因为安装程序中用了某种特殊的图像，碰巧我的机器不支持。换了台机器，安装后该程序运行得很好，但安装程序本身还是显示不全。同样地，厂家为了追求精彩，却适得其反了。

从这一节可以看出，研究一下别人封装产品的方法，能取得事半功倍的效果。我发现事先写下注意事项，在动手编写安装程序之前仔细阅读一遍，可以少犯许多错误（几乎没有人可以一蹴而就）。

16.2 收集文件

千万不要以为不经过研究，就可以在二、三秒内作好一个安装程序，多数情况下，Windows环境中的所有程序都比程序员预计的情况要复杂得多。例如：本书中的所有例子都依赖于C运行时库文件，到目前为止我还没有提到在应用程序中加入这些文件，源代码中也没有对它们的引用，这是因为第一次安装Visual C++时，已自动放在SYSTEM目录下，关联是自动进行的，在设计应用程序时不需要了解。对MFC文件和数据库应用涉及的文件情况相似，Visual C++即使没有完全地却也基本上帮你增加了所需的文件。然而，现在要做的是将应用程序完整打包提交给最终用户，这时你再也不能无视这些文件的存在了——

它们也是应用程序的一部分，必须放在打包之中。问题是如何找出应该包含哪些文件呢？

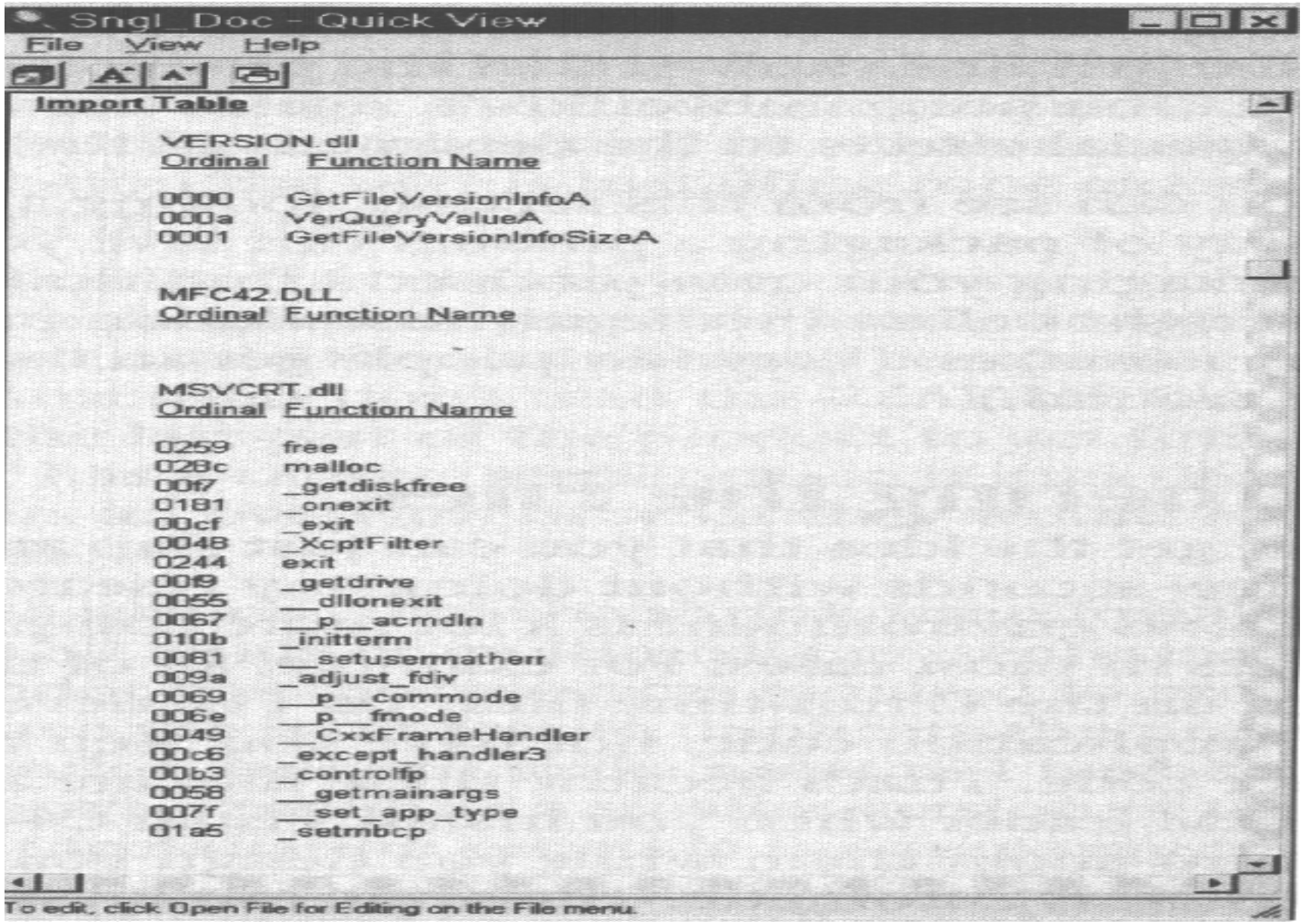
找出应用程序应该包含的所有 DLL 非常费时间，特别是当使用测试错误的处理方法时更是如此。然而，你可以使用如下的三种方法来进行测试错误处理：

方法一　使用 Windows QuickView 程序查看应用程序的 Import Table 表项，Import Table 列表列出了应用程序所有用到的 DLL。下页图（上）是我们在第 2、3、15 章介绍过的 Sngl_Doc 程序的 QuickView 程序列表，它不仅显示了该应用程序用到了哪些文件，而且给出了其中的函数。

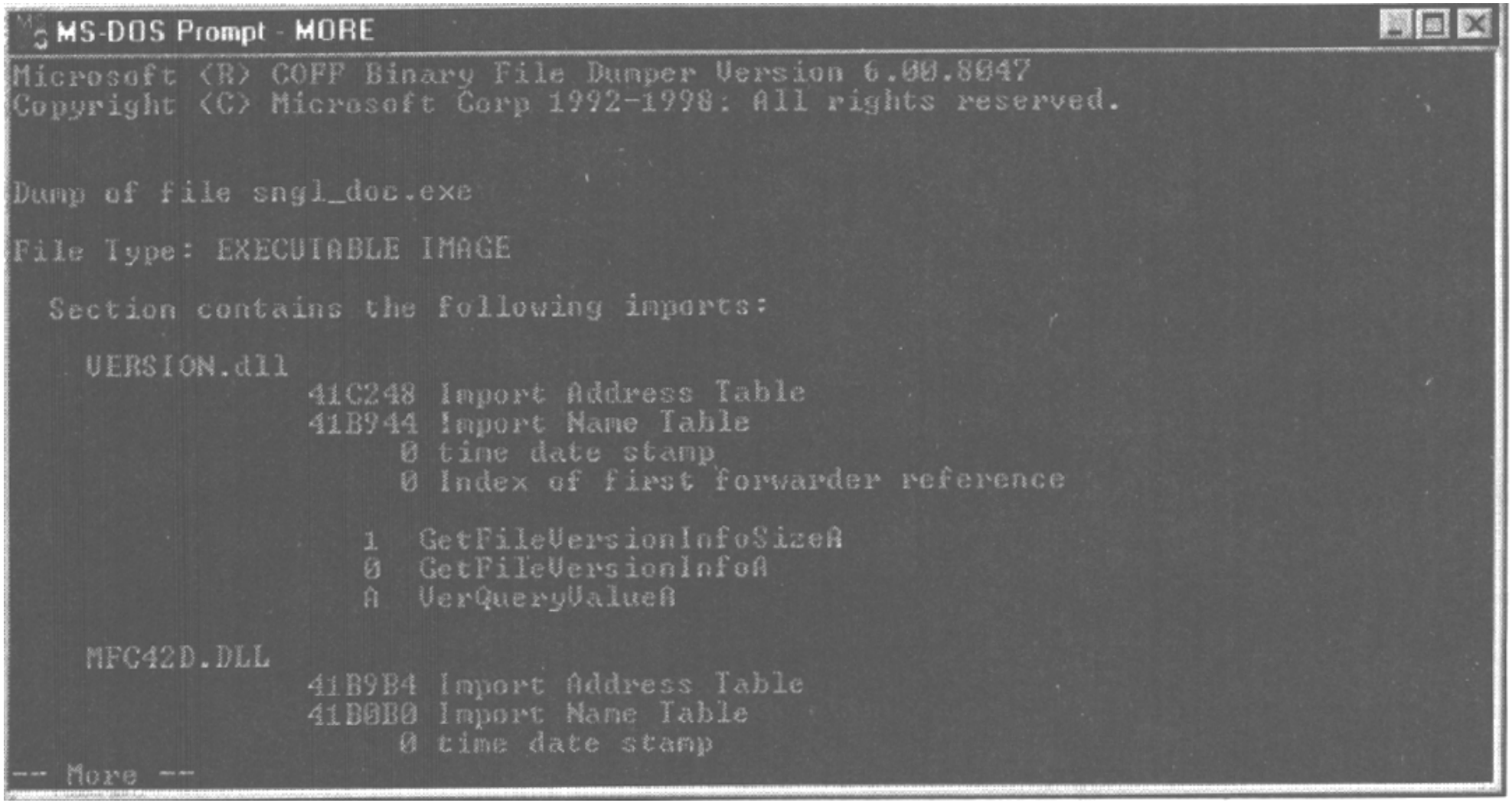
方法二　如果程序需要可以使用 DumpBin 程序给出 DOS 命令行方式显示的程序列表。用起来略微有点儿麻烦，但却常常可以发现 QuickView 发现不了的文件。而且它提供了更多的选项，可限定查找信息的种类，输入命令格式为：DUMPBIN /IMPORTS SNGL_DOC.EXE。下页图（下）是输出结果（这是个命令行程序）。

技巧　使用 DumpBin 实用程序时可能会遇到系统报出缺少文件的信息，如果使用的是安装时的缺省设置，那么 MSDIS100.DLL 和 MSPDB50.DLL 文件都在 Program Files/DevStudio/SharedIDE/bin 目录下，在 DumpBin 的执行过程中需要用到它和 VC/bin 目录下的 LINK.EXE 文件。

方法三　如果有最新版的 Windows SDK，就会看到 Depends 实用程序，用它就可以找出程序中用到的 DLL。


Sngl_Doc - Quick View
File View Help
Import Table
VERSION.dll
Ordinal Function Name
0000 GetFileVersionInfoA
000a VerQueryValueA
0001 GetFileVersionInfoSizeA
MFC42.DLL
Ordinal Function Name
MSVCRT.dll
Ordinal Function Name
0259 free
028c malloc
00f7 _getdiskfree
0181 _onexit
00cf _exit
0048 _XcptFilter
0244 exit
00f9 _getdrive
0055 __dllonexit
0067 __p__acmdln
010b _initterm
0081 __setusermatherr
009a _adjust_fdiv
0069 __p__commode
006e __p__fmode
0049 __CxxFrameHandler
00c6 _except_handler3
00b3 _controlfp
0058 __getmainargs
007f __set_app_type
01a5 _setmbcp
To edit, click Open File for Editing on the File menu.

找出这样一个文件清单还不是文件收集过程的结束，还需要找出相关的全部 DLL 文件。显然像 USER.EXE 文件这样安装 Windows 时就具有的文件不用包含进去，但必须找出那些没有的文件。总而言之，必须手工地检查每个文件及其相关文件，确保都包含在应用程序的打包中。



技巧 先仔细审查一下只装过 Windows 的机器中有哪些文件，如果以后要建立多个安装程序，这个文件清单就很有用了。

当应用程序文件清单完整之后，把它们放在一起，拷贝到装有更高版本的 Windows 的机器中，检测是否兼容。当缺少文件或是文件不在 SYSTEM 目录下时，就会看到警告信息。一定要保证放在 SYSTEM 目录下的文件必须到位，因为将来用户使用起来也是这样。

注释 尽量将全部 DLL 文件放在应用程序目录下，而不是在 SYSTEM 目录下，这样有助于删除程序的实现。而且会少受用户安装的其它程序的干扰，例如：所用到的某个 DLL 在其它程序中也用到了，而且是个老版本的，安装时会把你的新版本 DLL 给覆盖了。

16.3 建立安装程序

现在你已经清楚如何打包应用程序了，包括决定采用何种模式，比如：公司内部应用软件、共享软件和商业软件。还应该有一张所用到的全部文件的清单，并保证包括了每个文件安装所涉及的方方面面，例如：写出必须安装在 SYSTEM 目录下的文件。最后，应该在一台只装有 Windows 的机器上测试这些文件，直到通过之后，才能开始编写安装程序。

注释 本节内容以 Visual C++6.0 的 InstallShield 为基础，老版本的 Visual C++中的 InstallShield 版本太低，使用时必须修改某些过程

才能满足需求，即使你所使用的 InstallShield 与本书所用的 InstallShield 版本相同，你所看到的屏幕显示和选项也会因所封装的程序不同而有所差异。简而言之，使用时看到的和叙述的有些差异无关大局。但必须安装 InstallShield 产品，它在 Visual C++ 6.0 CD 的 Ishield 目录下。我们假定你已装好了。

到目前为止，我们已向第 2 章建立的应用程序 Sngl_Doc 中加入了一个资源文件（第 3 章）和一个帮助文件（第 15 章），可以进行打包提交给用户了，它至少可以作为一个例子。下文给出了建立一个典型安装程序的步骤，这些没什么费解的，就是为了程序的发布。如果非要从中提取出一个模型，近似于我们前面提到的企业内部应用程序型的打包情况。

注 本节讲述完整的安装程序，但实际应用中必须进行修改以满足特定的程序和安装环境的需要。

1. 启动 InstallShield 6 Free Edition 程序。你会看到如图 16.1 所示的窗口，注意：事先并没有定义工程文件，但在 Projects 窗口中出现了一个安装向导，而且看上去不像 Visual C++中的标准窗口，但随着工作的进行，你会发现它非常有助于安装程序的建立。同时请注意一下右下角的 InstallShield 链接，单击将打开浏览器，打开 Internet 上的 InstallShield 在线帮助站点。

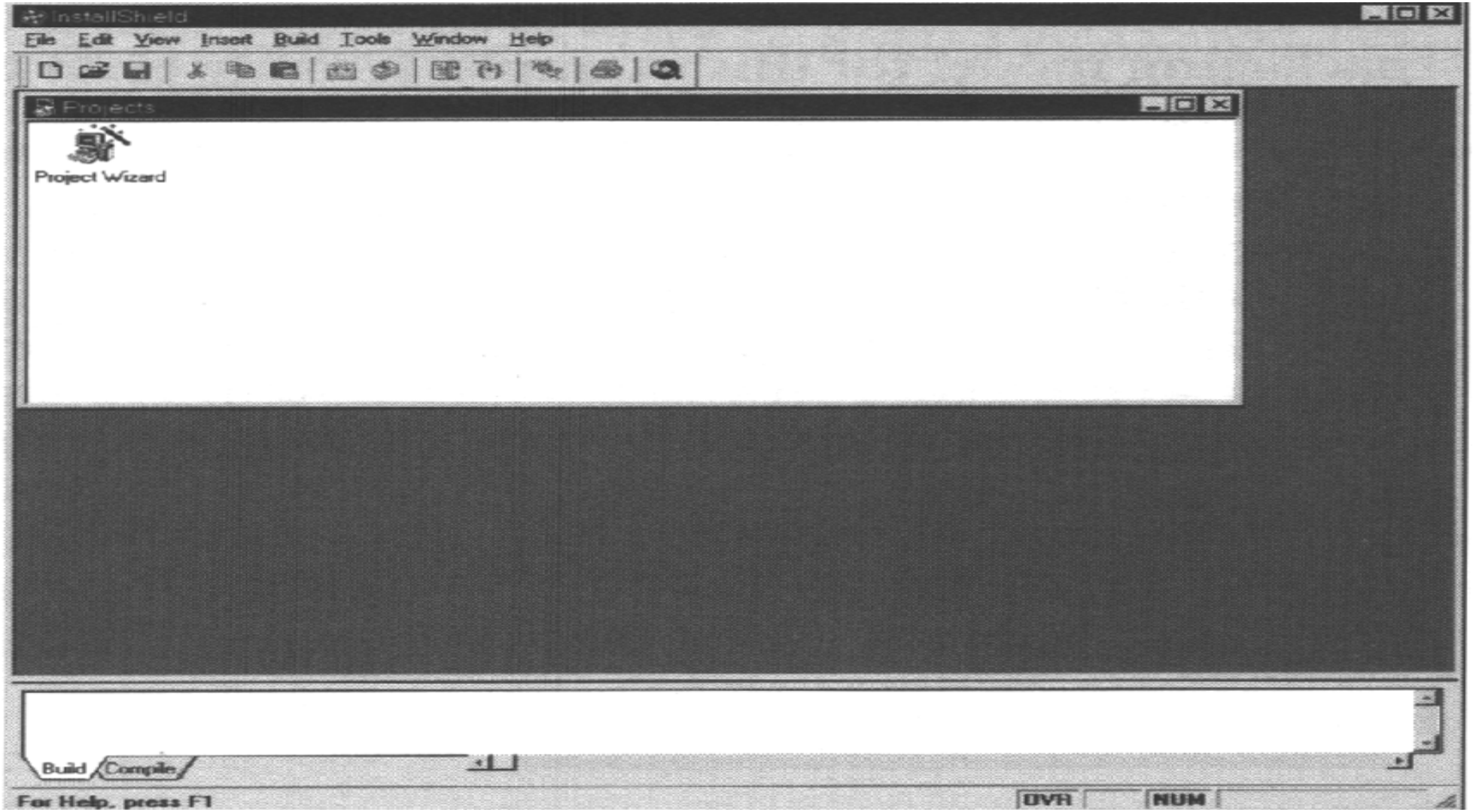
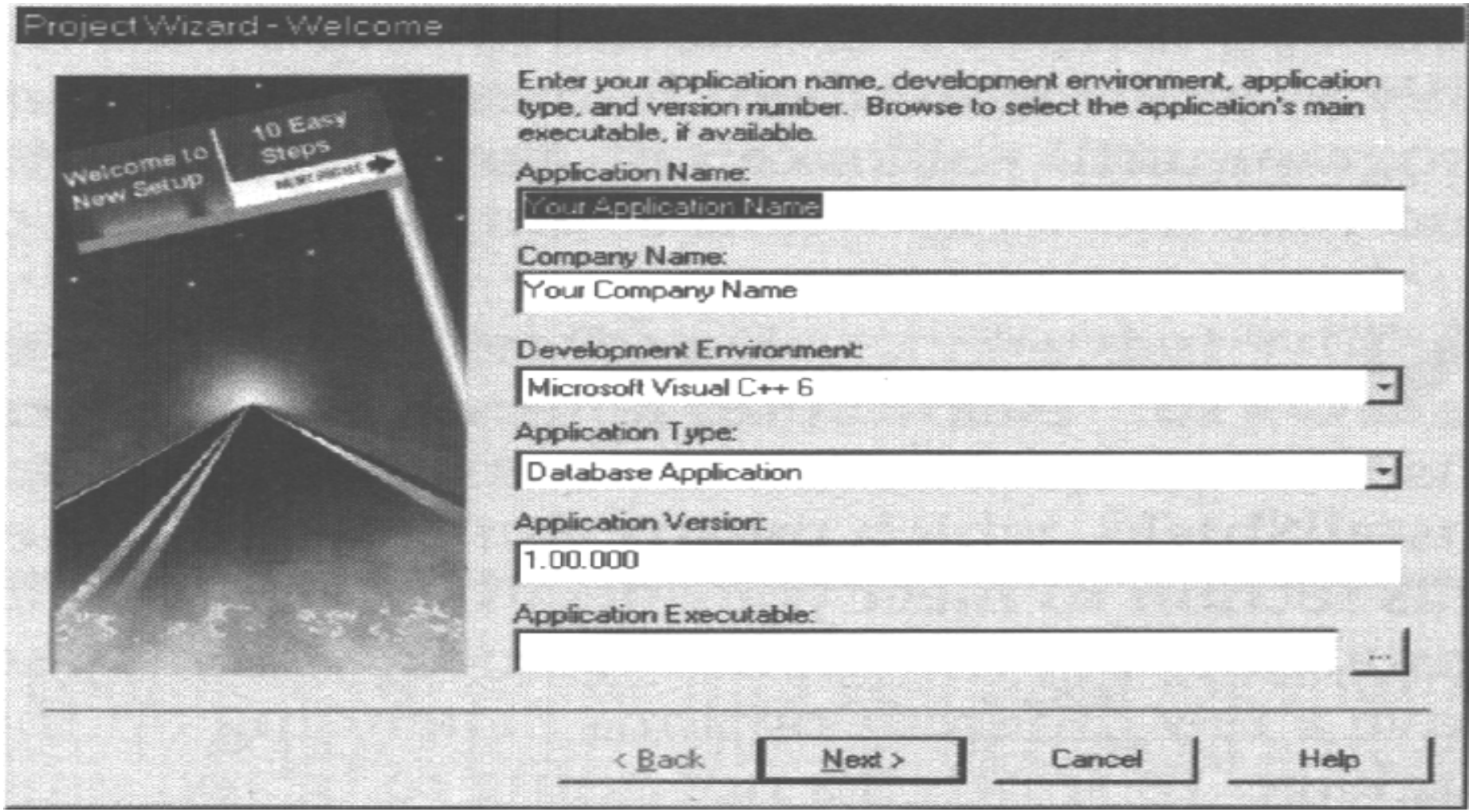


图 16.1　InstallShield 让你使用与 Visual C++中创建应用程序所用技术相似的技术创建安装程序

2. 双击 Projects 窗口的 Project Wizard 图标，你会看到如下图所示的 Project

Wizard-Welcome 对话框：



注　InstallShield 的 Professional Edition 在 Welcome（及其它）页提供了高级特性——请一定要读一读帮助文件的增强特性部分。

3. 输入应用程序的名称，示例程序使用了 Single Document Application

Example。

4. 输入公司名称，示例程序用的是 A Sample Company。

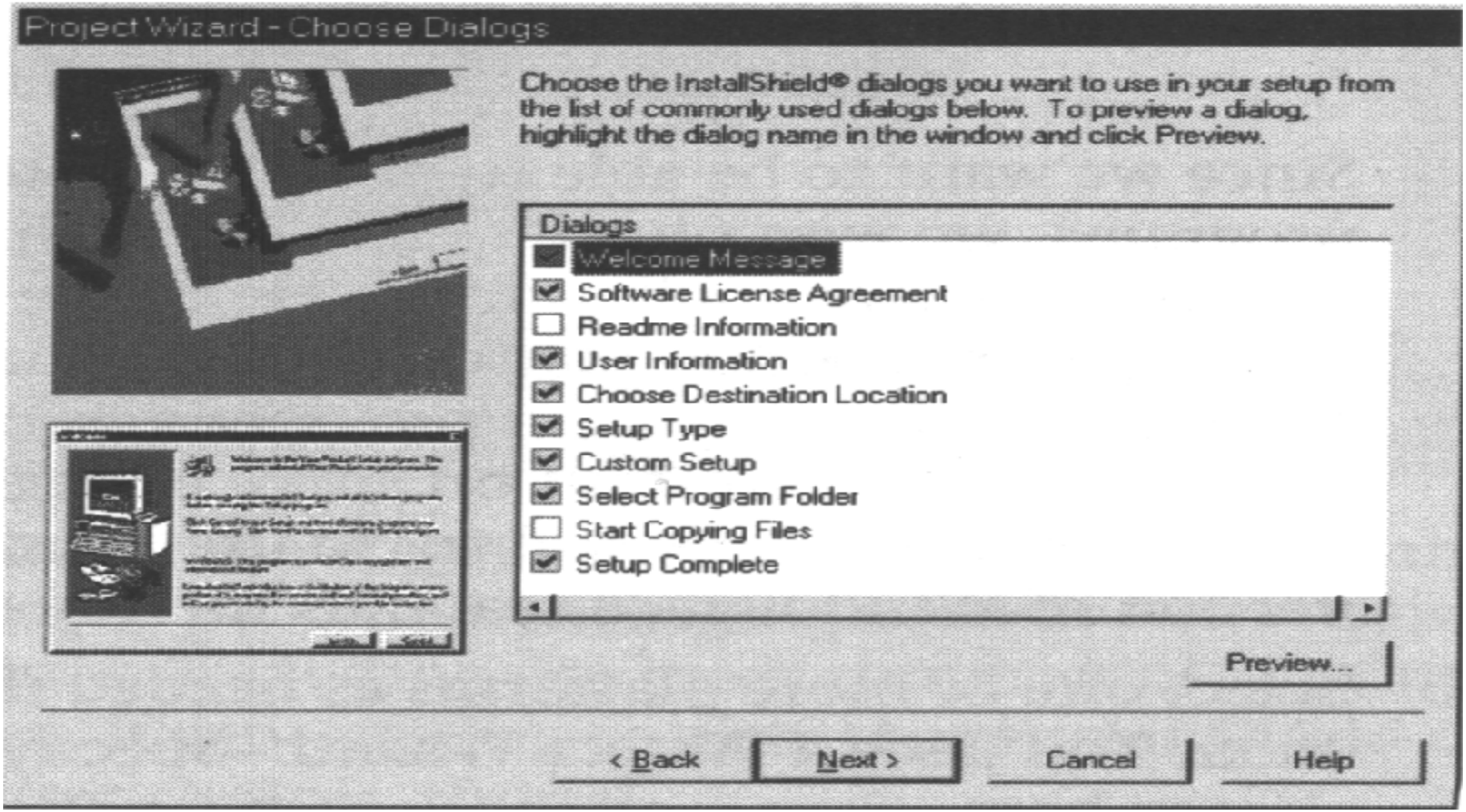
注　InstallShield 的 Free Edition(免费版)只为 Development Enviroment 域提供了一个选择项：Microsoft Visual C++ 6。

5. 在 Application Type 域选择一个列表项，示例程序选的是 Software Development Applicaion；该安装程序没有标准程序类型，而 Software Development Applicaion 支持实用程序。

6. 在 Application Version 域输入版本号，我们用的是 1.0，当然可以沿用自己公司中的编号。

7. 单击 Application Executable 域旁边的“...”（省略号）按钮，你会看到一个标准打开对话框用于查找硬盘上的应用程序，示例程序用的是 SNGL_DOC.EXE(我们在第 3 章中建立、第 15 章中修改的那个程序，在 Resource 文件夹下）。

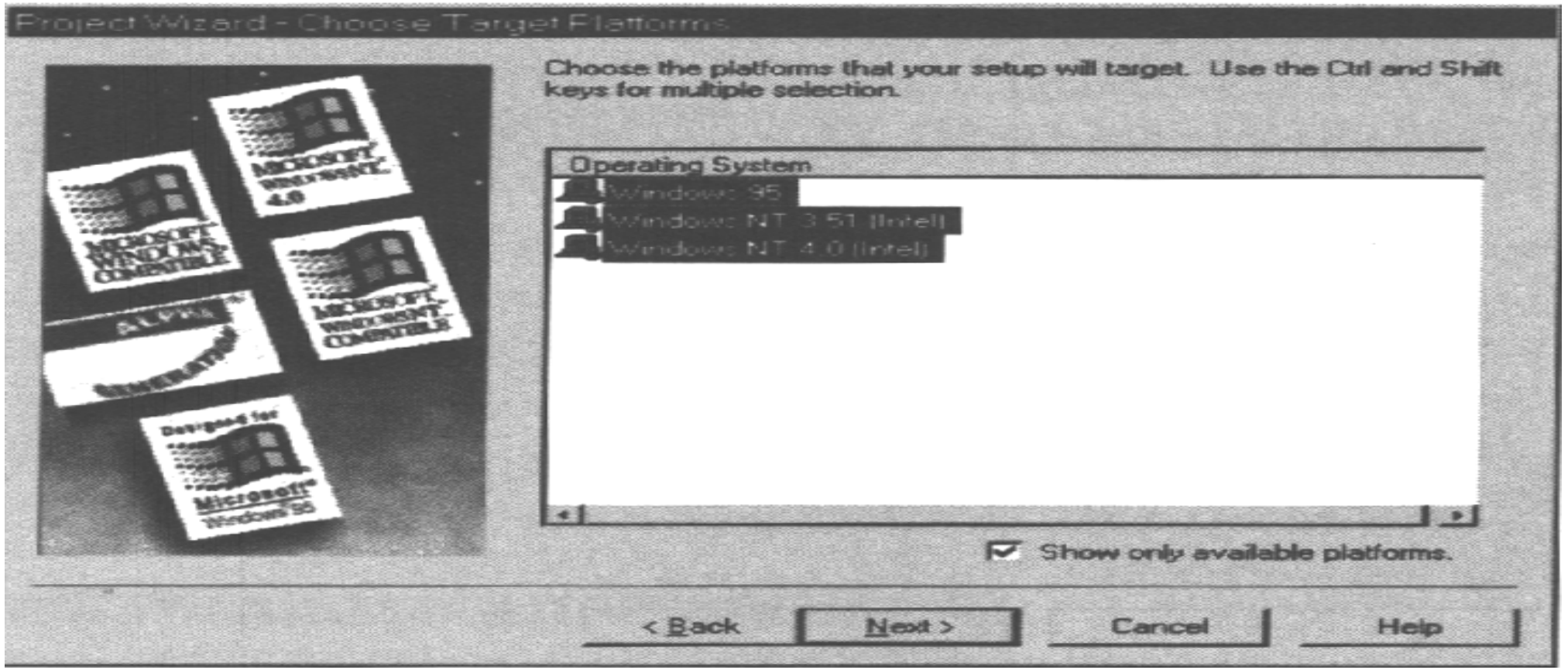
8. 单击 Next，你会看到如下图所示的 Project Wizard - Choose Dialogs 对话框：



在这里选择的一系列对话框会在用户的安装过程中显示出来，因为我们使用的是企业内部应用型的程序，所以不选中 Software License Agreement、Setup Type 和 Custom Setup 对话框。对本章前面部分讨论过的其它打包模型来说，则可能需要选择其中的某些项目。在这个对话框中有几个需要注意的地方：首

先，当从一个对话框选项转移到另一个对话框选项时，请注意该对话框左下角的变化，Project Wizard 会显示出该对话框的一个微型缩影，以方便你确认是否要选用这个对话框；加亮一个对话框后单击 Preview 按钮，所看到的对话框与用户看到对话框相同。显然，你以后还可以修改这个对话框，但预览方法有助于你预先了解安装程序中要使用的对话框。

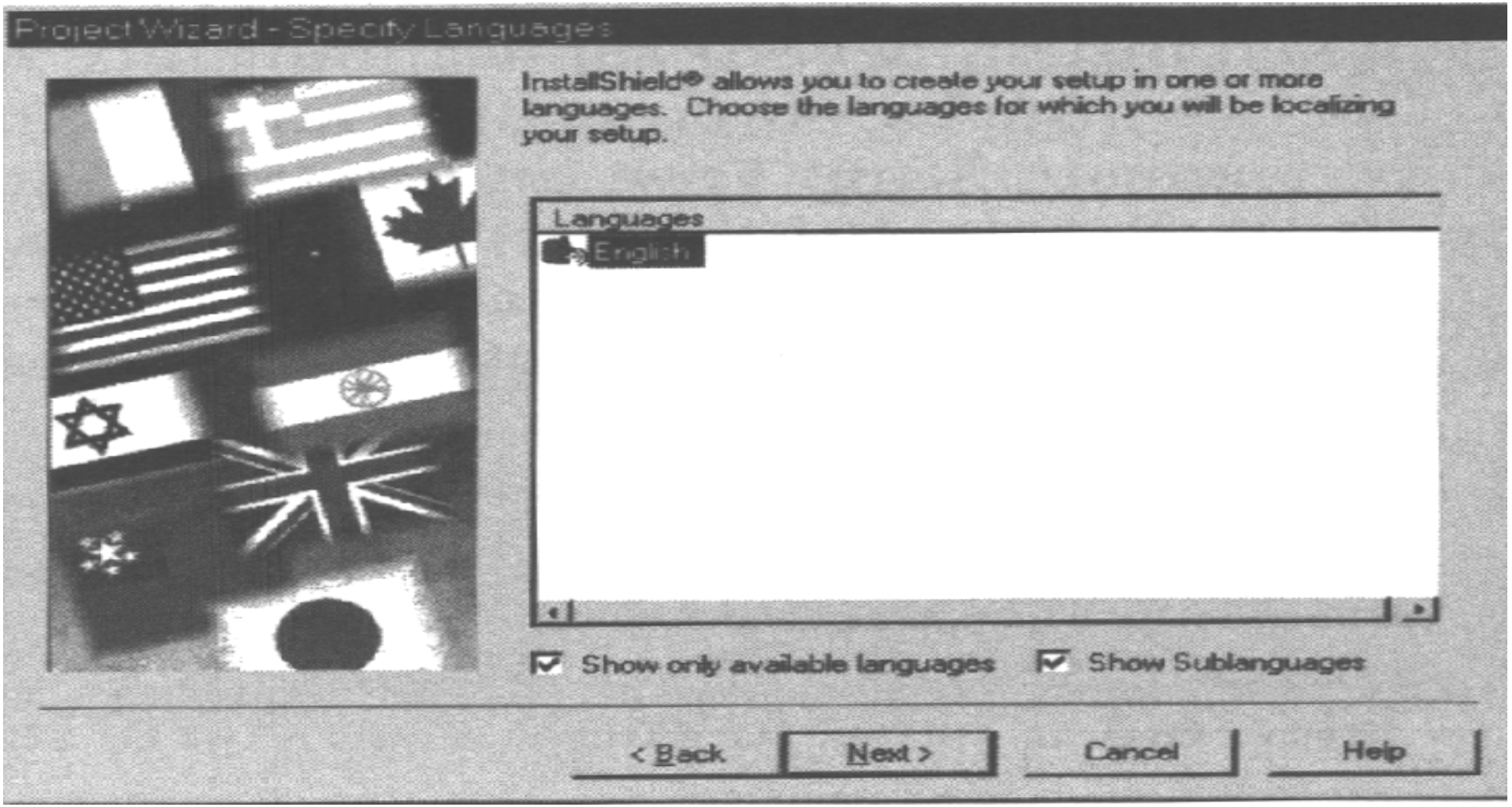
9. 选择想要加进安装程序的对话框，单击 Next，你会看到如下图所示的 Project Wizard - Choose Target Platforms 对话框：



由于我们想把我们的程序安装到各种支持程序上，因此不需要改动要安装的平台。然而，在某些情况下你可能想作些选择。减少支持平台的种类会同时带来两种结果：第一、减少了所建立的安装程序的大小——这对于开发占用空间少的共享软件很有帮助；第二、减少了用户在不适用的 Windows 版本中使用程序的可能性——这对所有程序设计者都有帮助。

注 不选中 Show only Available platforms 复选框时，系统将会显示专业版 InstallShield 提供的所有平台选项。

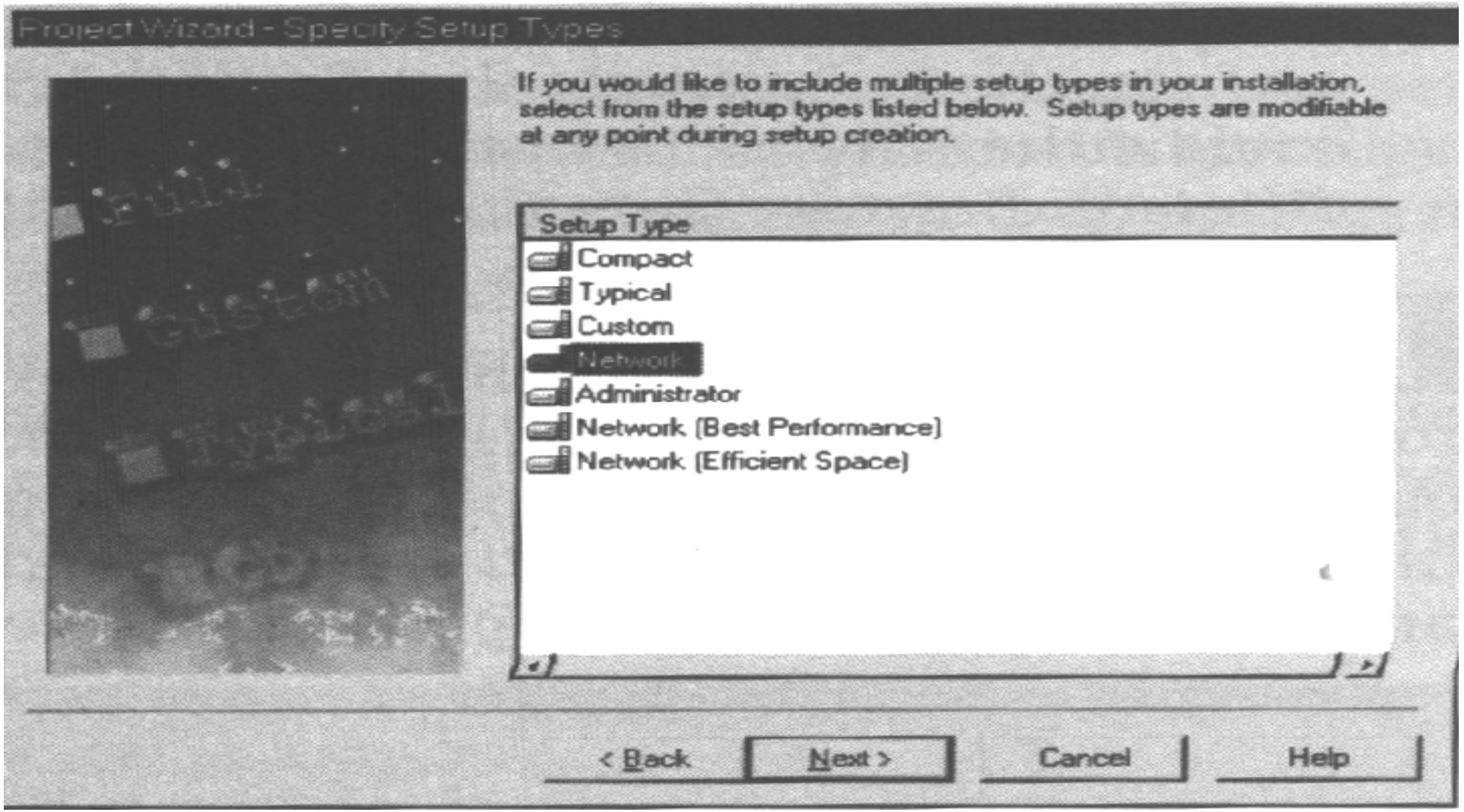
10. 根据需要选择一种或多种（至少一种）平台，单击 Next，你会看到如下图所示的 Project Wizard - Specify Languages 对话框：



由于 Free Edition 版只支持一种语言，所以这里不需要作改动。与所有其它的选择一样，多选并没有必要。增加语言必然增长安装程序的长度，也会给用户造成困惑，最好只选所用的语言。

11. 选择所用的一种或多种（至少一种）语言，单击 Next，你会看到如下

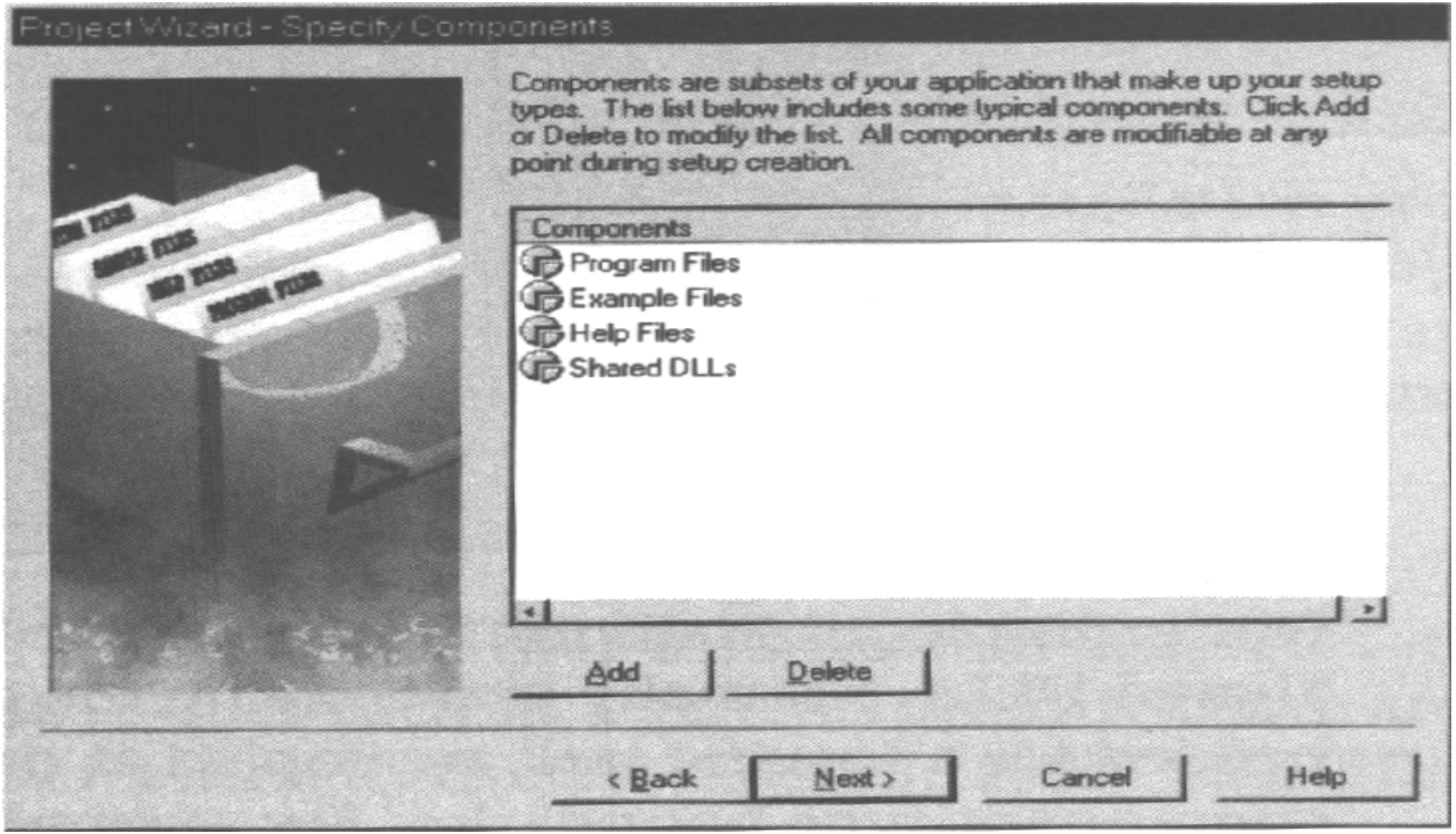
图所示的 Project Wizard - Specify Setup Types 对话框：



因为我们使用的是企业打包模型，所以我选择了 Network 安装类型.其它打包模型要求作出其它选择，包括常见的方式：定制方式、典型方式和最小方式。

有趣的是 InstallShield 还提供了一些其它的选择。

12. 选择一种或多种安装类型，然后单击 Next，你会看到如下图所示的 Project Wizard - Specify Components 对话框：



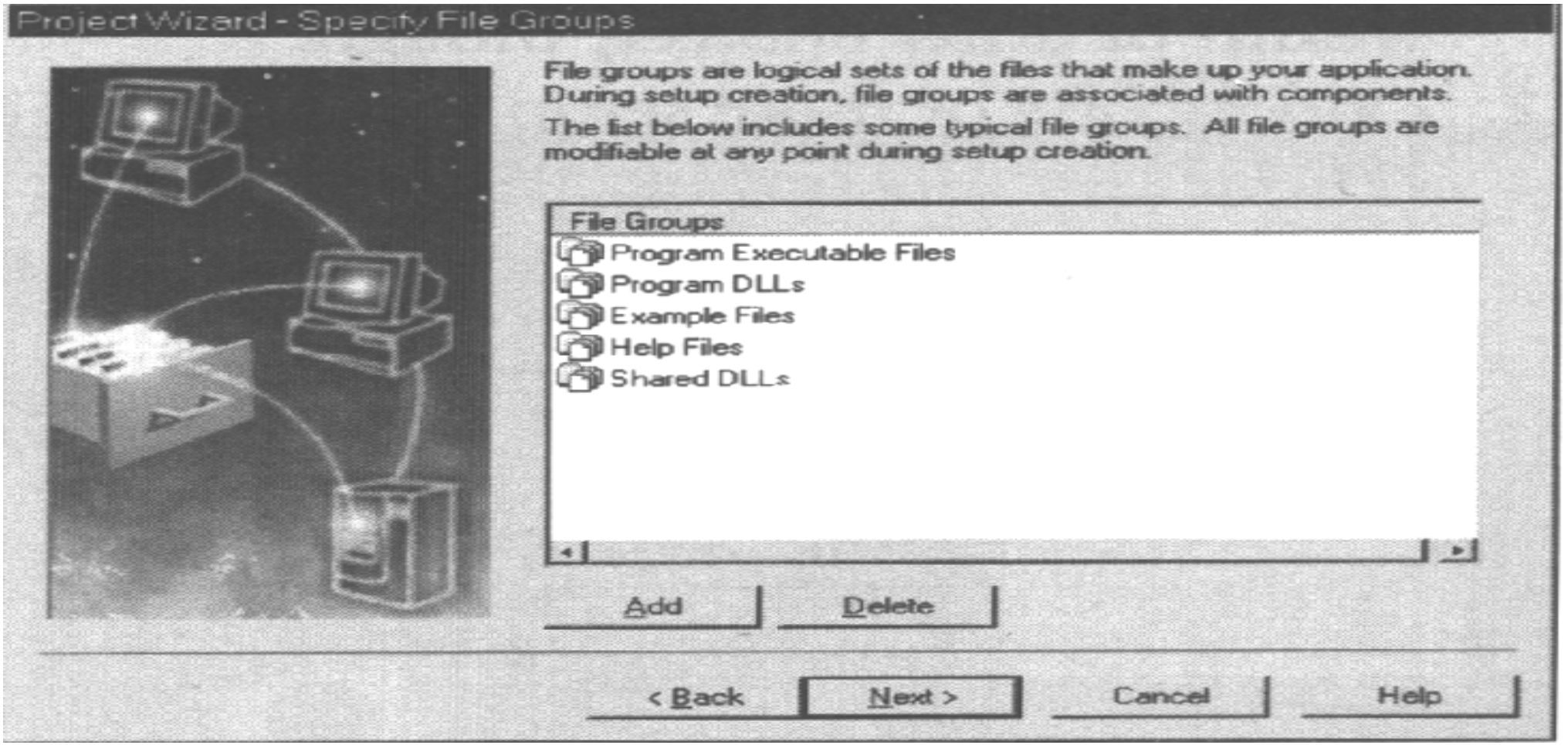
Project Wizard 并不知道你打算安装哪些组件。示例程序使用了 Program Files、Shared DLLs 和 Help Files。你需要每次选择一个组件类型，并定义该组件类型下的文件。组件类型对程序进行了主要部分的划分，例如，某个用户在安装时希望安装所有的示例文件，但不安装帮助文件。组件类型并不需要指出具体由哪些文件完成该组件的任务，而是让用户选择把不同组件中的文件安装在同一目录或不同的目录下。

技巧 可以根据需要增加新的组件类型。例如：如果是数据库应用程序需要加入 Database File 组件，作起来很简单：单击 Add 按钮，向导会自动加入新的列表项，输入组件名称，按 ENTER 键就行了。

---

13. 选取 Example Files，单击 Delete，Project Wizard 会删除该组件类型。

14. 根据需要增加、删除组件类型，单击 Next，你会看到如下图所示的 Project Wizard - Specify File Groups 对话框：

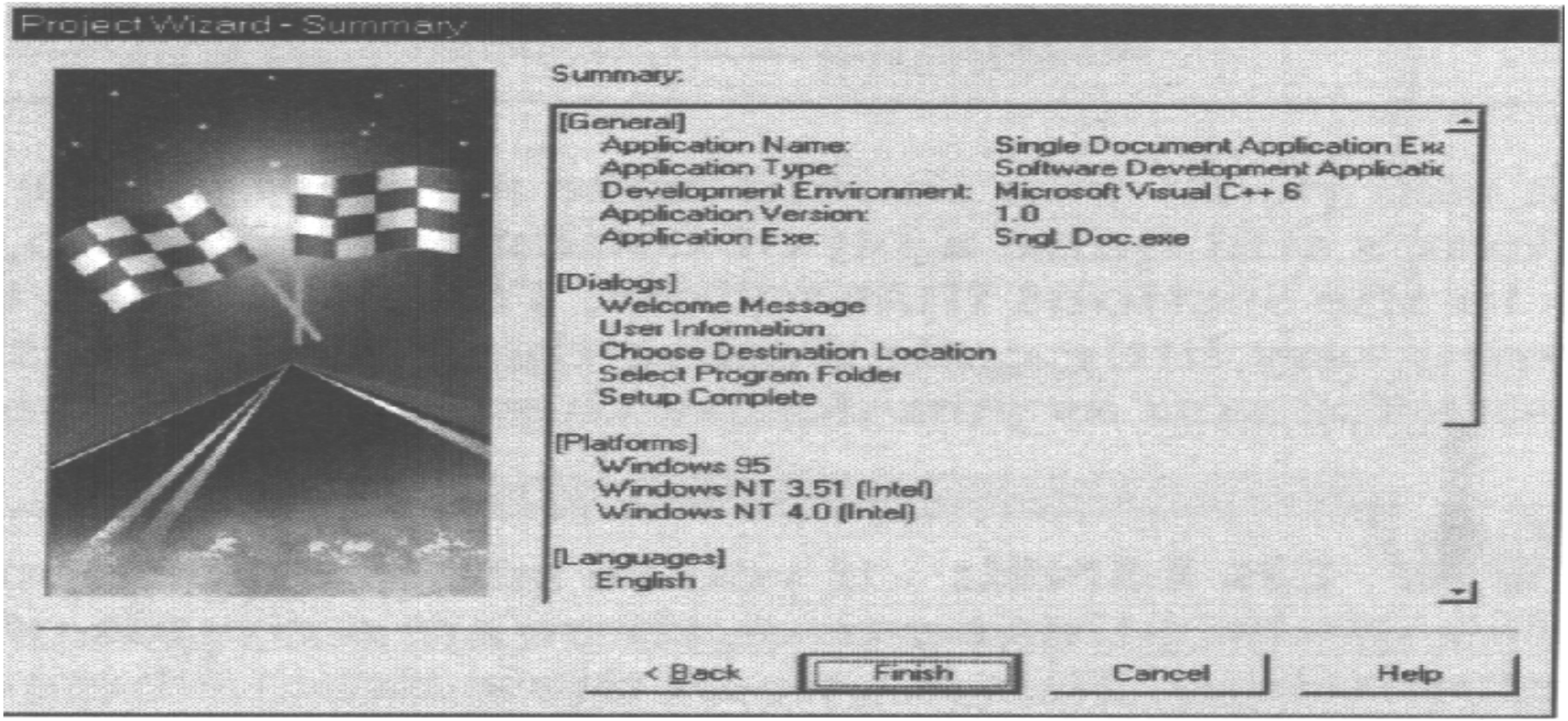


示例程序使用了 Program Executable Files、 Help Files 和 Shared DLLs。正常情况下，你会创建一些文件组，以便完成特定任务所需的所有文件能一次拷贝到目的地。文件组不受组件类型的限制，例如：拼写检查和语法检查都依赖于同样的 DLL 文件集，但使用的规则不同，你可以将 DLL 定义为一个共用的文件组，规则文件分别定义为另外两个组。如果用户选择了拼写检查，InstallShield

就会拷贝相应的规则组和通用 DLL 文件组，但不会复制语法检查文件组。

15. 选取 Program DLLs，单击 Delete，选取 Examples Files，然后单击 Delete，删除本示例程序中不需要的两个文件组，显然文件组的选择取决于程序的组织方式、所需文件的复制位置。请谨记，同一文件组中的所有文件都将拷贝到硬盘的同一个目的目录下。

16. 根据需要增加、删除文件组类型，单击 Next，你会看到如下图所示的 Project Wizard - Summary 对话框：



这时应检查所列清单，保证正确无误，然后再开始生成安装程序。

17. 单击 Finish。InstallShield 用所选定的选项生成安装程序，界面会变为如图 16.2 所示的样子，注意这时你会看到创建安装程序所需的 C++源代码，你可以像修改其它工程那样对这个工程进行修改。

注释 不要以为 InstallShield 生成的源代码文件与普通工程生成的源代码文件完全一样，实际上，它生成的文件中有些是你从未见过的辅助性文件，比如安装规则文件（SETUP.RUL），如图 16.2 所示。

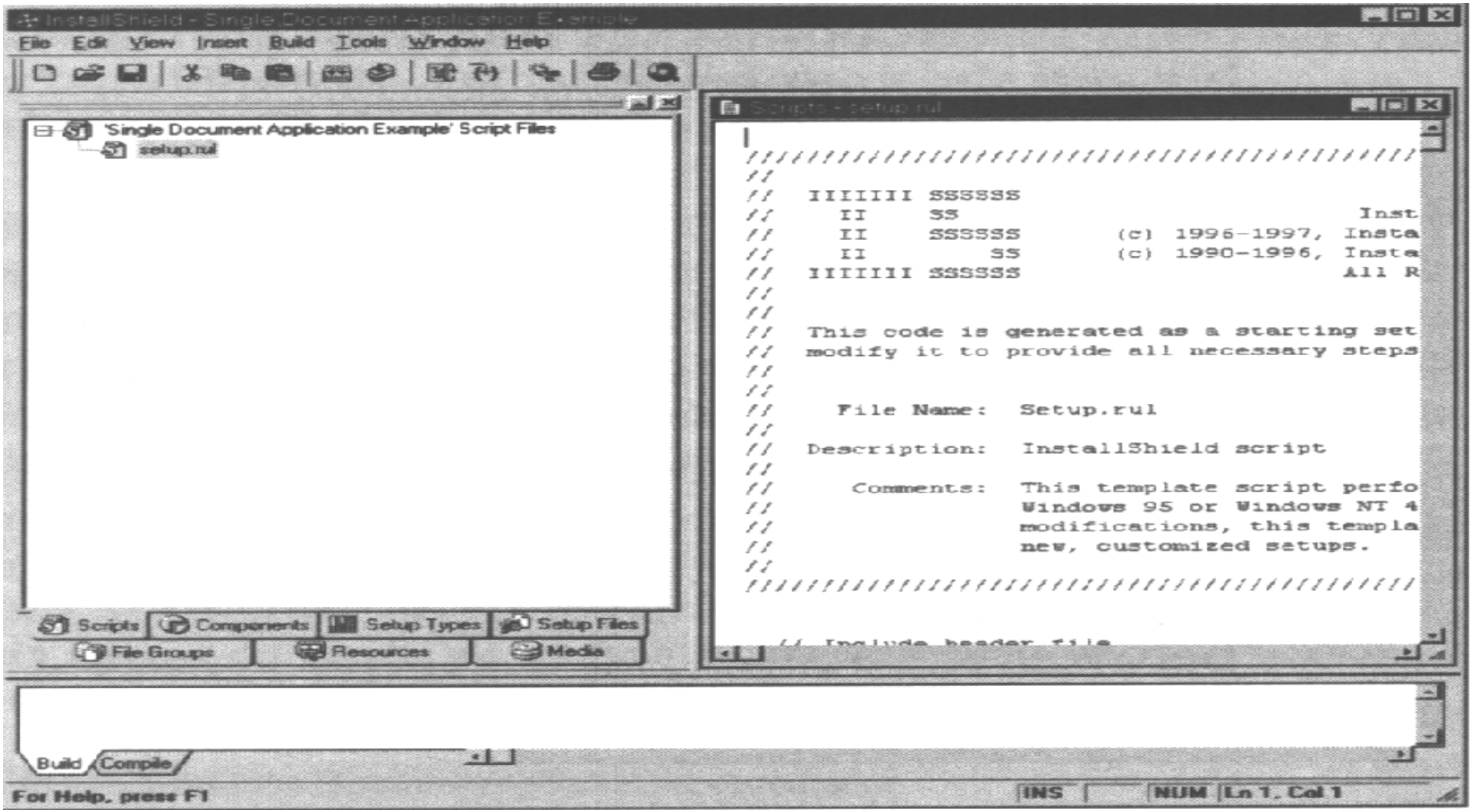


图 16.2　建立了工程文件之后，InstallShield 会自动生成并显示结果源代码

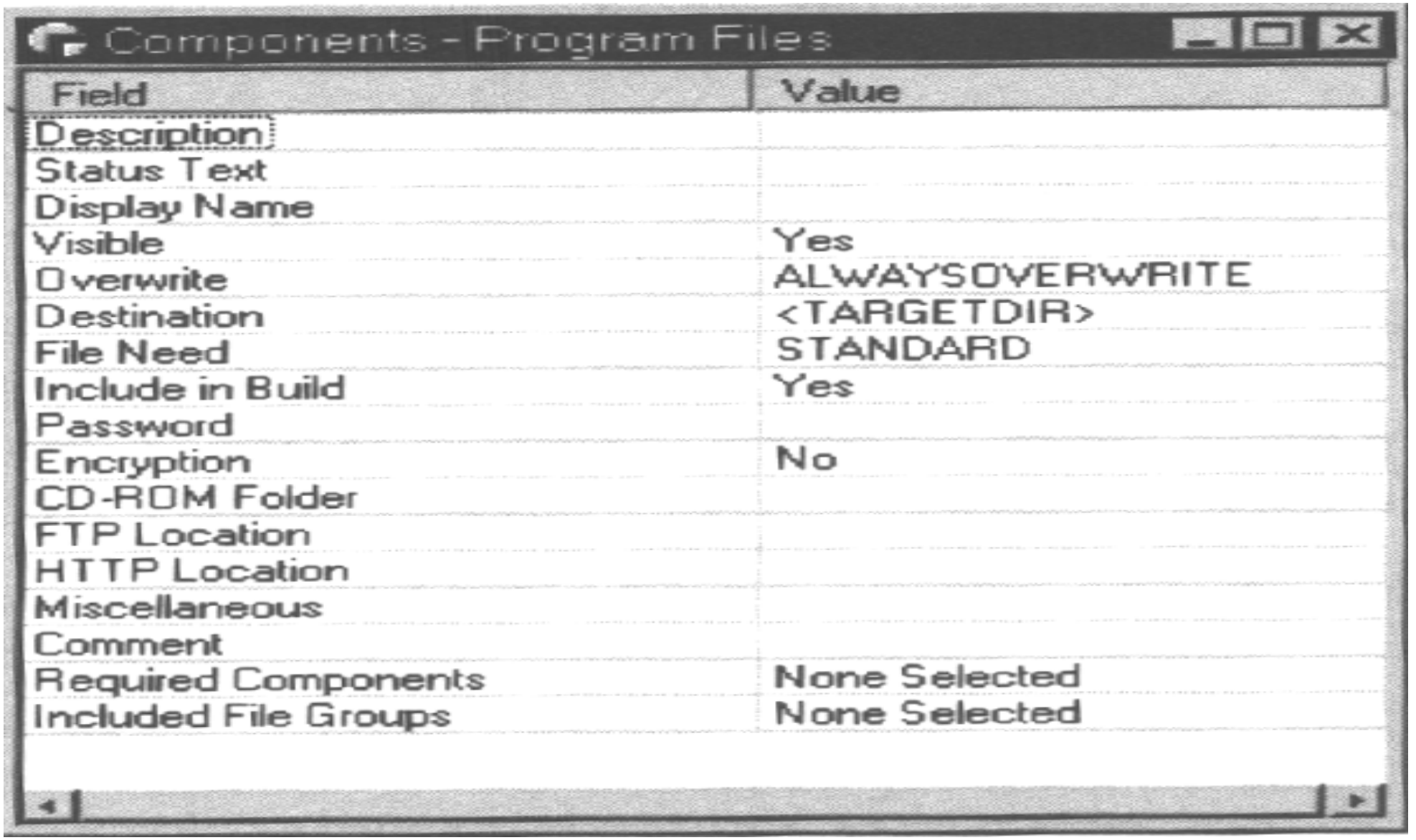
这时，就有了一个空白的安装程序外壳，还需要作很多的配置，我们将在下节阐述。最重要的是要看到整个过程中几乎没有写什么代码，在定义了各种安装程序元素之后 InstallShield 就替你实现了。

Web 链接 如果使用 InstallShield 还有问题，有许多方法可以得到帮助。这时最需要读一下 README 文件，其中含有许多 Internet 地址，例如：新闻组站点 http://support.installshield.com/newsgroups/default.asp，它链接到了几个 InstallShield 专题的新闻组，只需单击一下，在你的新闻阅读器中就会建立一个新的文件夹。甚至对 Free Edition 版也有专门的新闻组，地址为 installshield.is5.free-edtion。还可以找到讨论 IDE、脚本和多媒体用法的新闻组，总而言之，这些新闻组提供了与其它程序开发者交流使用打包程序、成功打包应用程序的经验。

设置组件

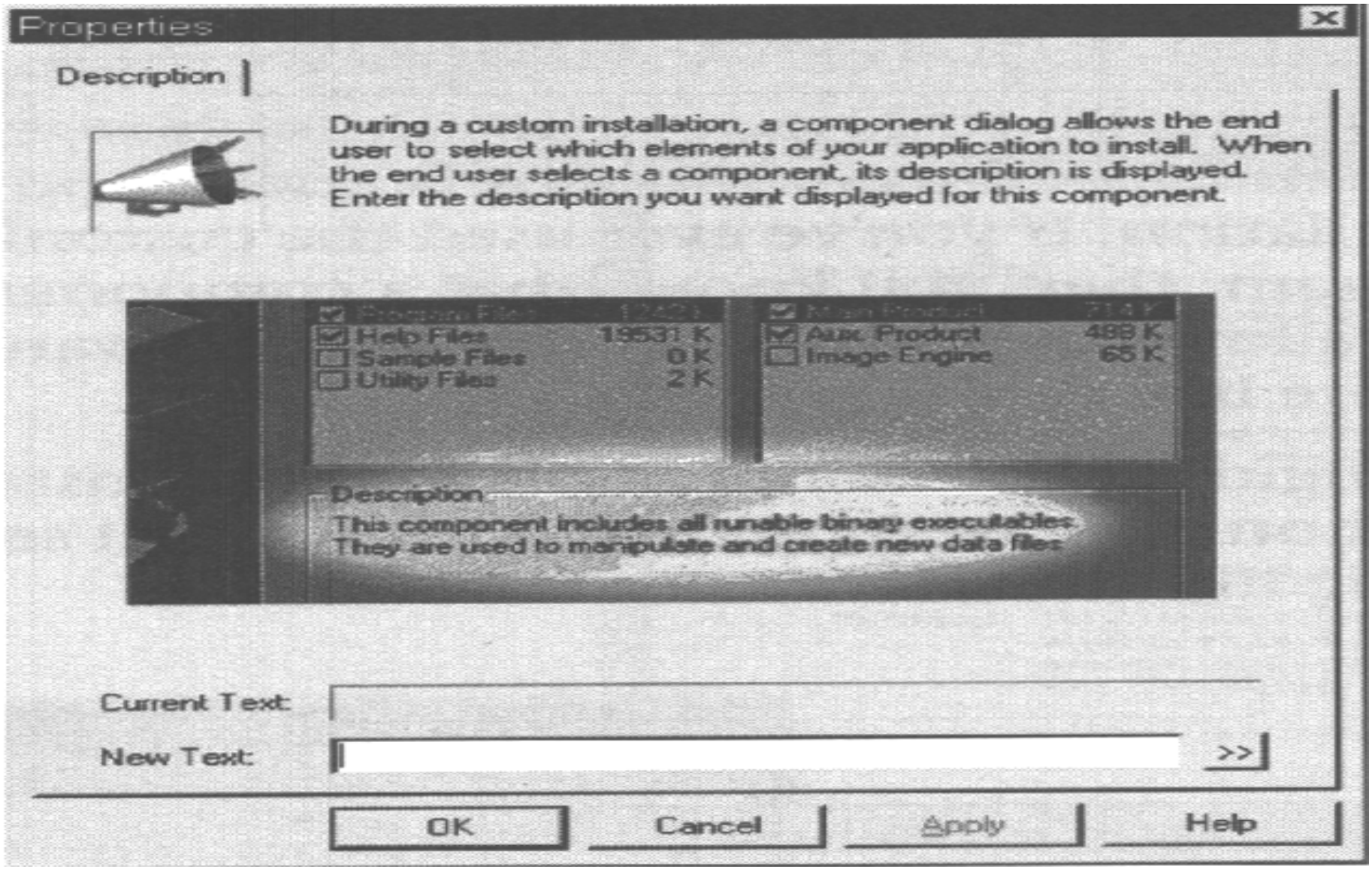
首要的任务是设置各种组件。组件是安装过程中出现的选择项，如果以前用过定制安装，就会很清楚。它们是一系列的复选框，这些复选框让你选择是否安装某个程序项。

配置组件相对比较容易，单击 Components 选项卡，你会看到如下图所示的 Components - Program Files 对话框：



组件对话框中包含了所选组件的完整清单，这里有三个组件：Program Files、

Help Files、 Shared DLLs。在阅读下一节之前，必须全部设置这三个组件。双击 Description 属性，你会看到如下图所示的 Properties 对话框：



其它所有的属性对话框与该对话框相似。每个属性对话框都用来描述要用该属性完成什么任务。也可以输入属性的值，本例中输入为：All the files required to run the application。完成后单击 OK，Description 属性就包含了所输入的内容。

不需要改变所有的组件属性——真正修改你进行的修改取决于所建立的应用程序的打包类型，例如，对于公司内部应用程序，一般不允许用户选择要安装的组件，所以没必要定义组件描述。但是，无论选择何种打包模型，有些属性都需要修改，下面我们逐一讨论：

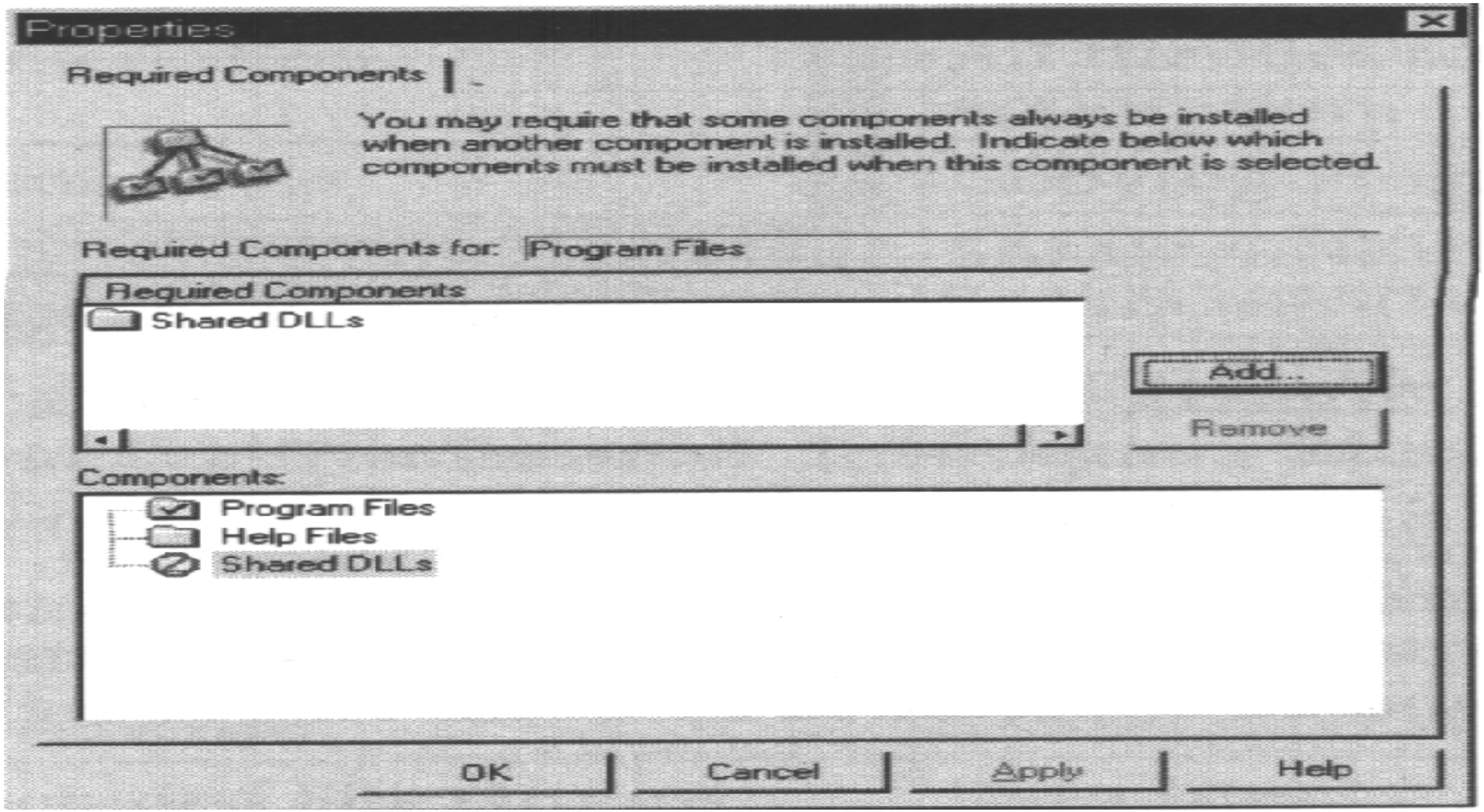
Status Text（状态文本） 这些文字是在安装程序向目标盘拷贝文件时用户看到的提示。Progress 对话框将显示一些类似于“Copying program files...”的提示，这些提示适用于缺省组件。对定制组件来说，你或许想给出一些特殊信息。

Installation（安装） 如果遇到过自己的最新版本的 DLLs 被其它程序覆盖的情况，你就会切身体会到这个属性的重要性。该选项告诉 InstallShield 在覆盖文件之前你是否想先检查它的时间戳，我总是使用 NEWERVERSION/NEWERDATE 或 SAMEORNEWERVERSION /SAMEORNEWERDATE 选项来代替缺省的 ALWAYSOVERWRITE 规则。实际上，这些选项告诉 InstallShield 仅当所装文件比硬盘中现有文件日期更新或版本更高时才覆盖原文件。

Destination（目的） 所有应用程序文件的标准安装位置是目的目录，它由用户选择。然而有些情况下使用用户设定的目录会浪费盘空间，例如：大多数 Visual C++程序运行时要求 C 运行时库文件和 MFC 文件，

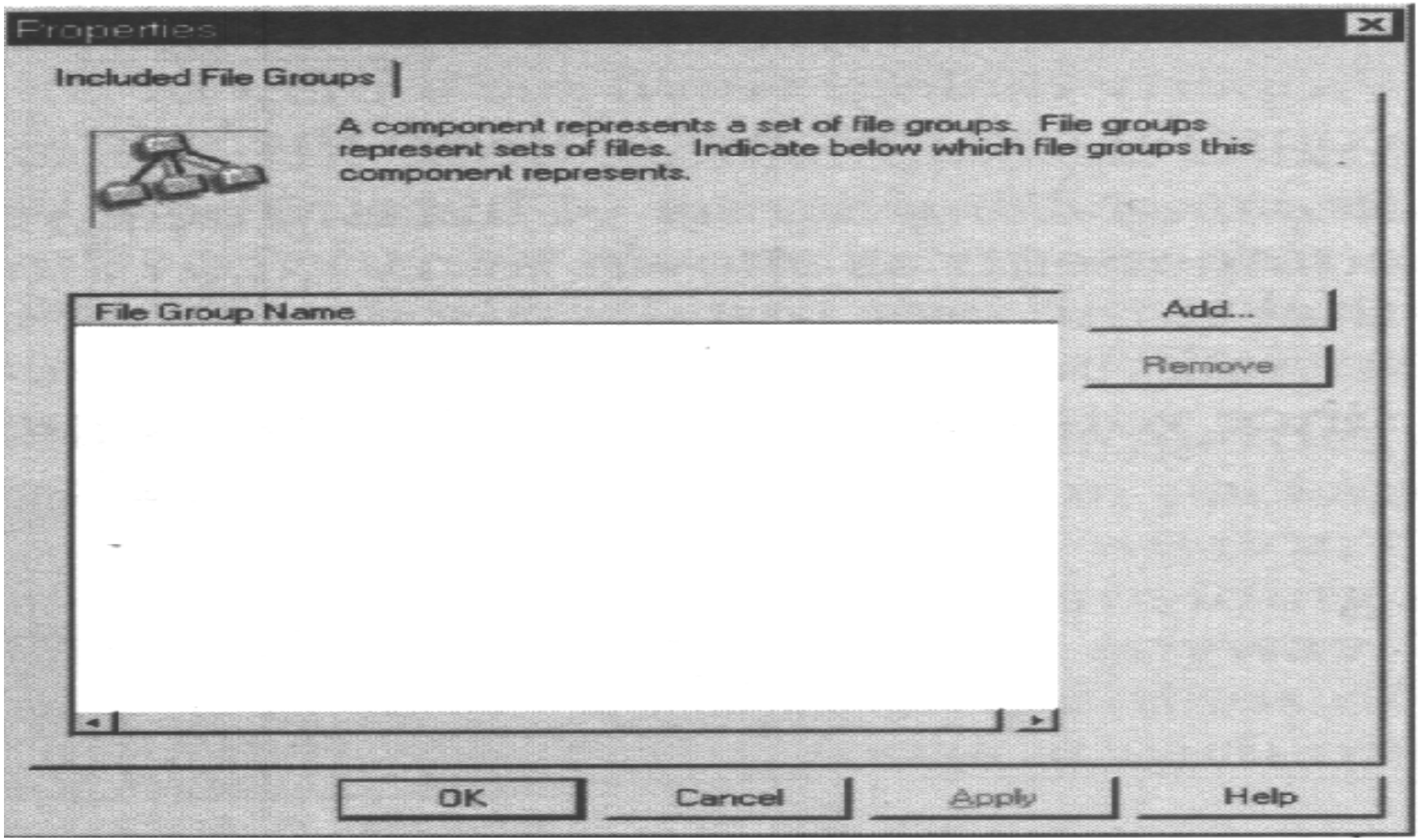
如果每个应用程序都把它们打包后安装到自己的目录下，硬盘空间该是多大的浪费啊？因此，总是把这些共享的 DLLs 放在 Windows SYSTEM 目录下，这时就要更改这个属性（打开属性对话框时，会看到可供拷贝的所有映射地址，选中需要的那个）。

必需的组件　在这里设置组件间的关系，如下图所示。图中我告诉 InstallShield，当用户安装 Program Files 组件时，他或她也必须安装 Shared DLLs 组件（只有在没有 Shared DLLs 组件程序就不能正常运行的情况下定义组件间的关系才有意义）。

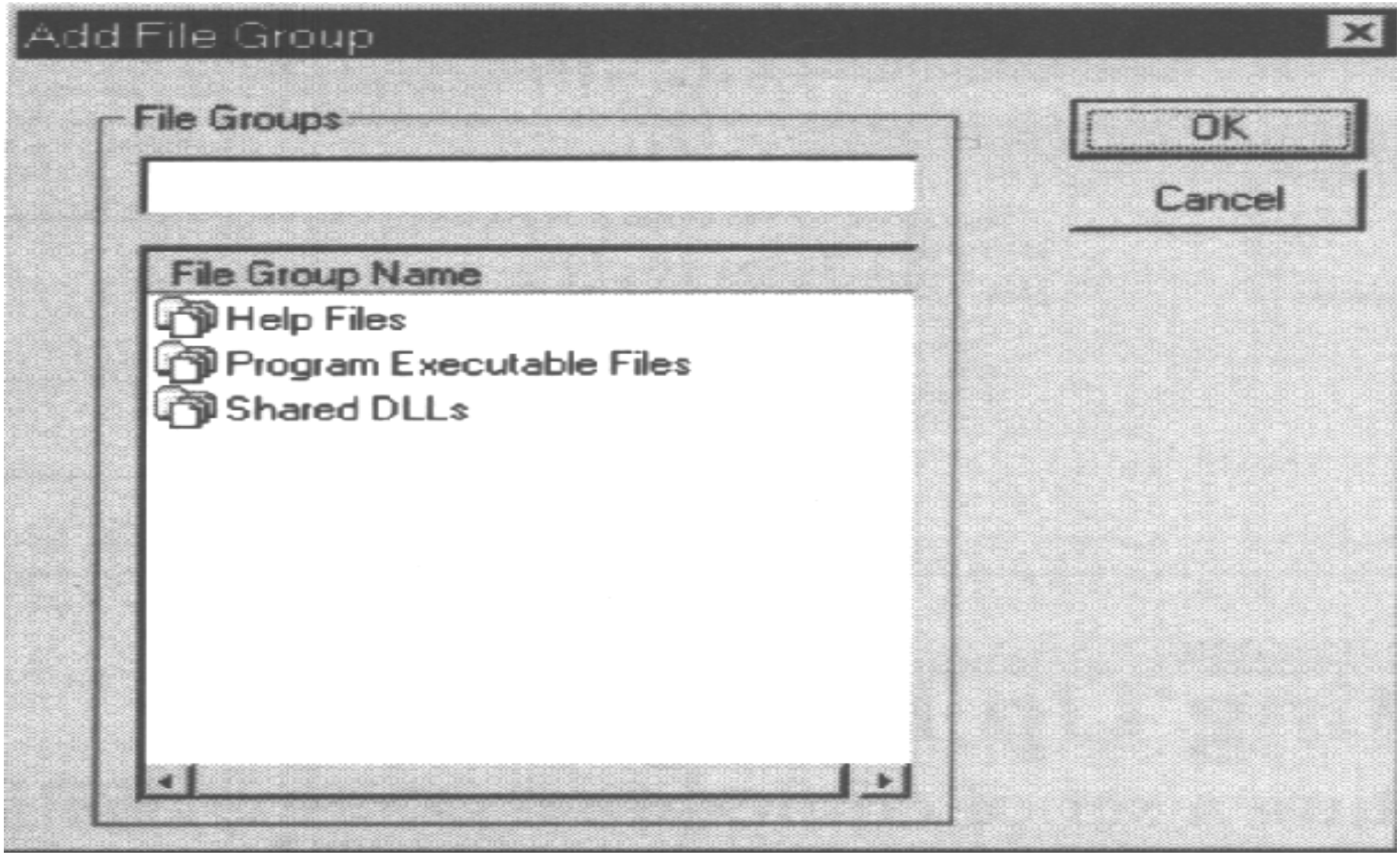


技巧 如果你打算在 Internet 或 Intranet 上安装应用程序，那么请特别留意 FTP Location 和 HTTP Location 属性。这些属性使用户可以从 Web 链接开始安装，并从 Web 服务器上拷贝所需的文件。如果安装文件处于 FTP 服务器的控制之下，那么一般选用 FTP Location 属性。

配置组件时最后一个绝对必须留意的属性是 Include File Groups 属性，直到现在我们还没有给 Program Files（或其它）组件指定任何东西。双击该属性，你会看到如下图所示的 Properties 对话框：



Properties 对话框包含了指定给某个组件的所有文件组的清单，前面我们讲过一个例子，安装时可以选择用拼写检查还是用语法检查，它们的 DLL 文件相同，规则文件不同。前面我还讲到过要设置三个文件组：一个带有语法检查规则文件，一个带有拼写检查规则文件，第三个带有通用的 DLL 文件。现在就是设定其间关系的时候了，单击 Add 按钮，你会看到如下图所示的 Add File Group 对话框：



正如你所看到的，Add File Group 对话框包含了我们使用 Project Wizard 定义的三个文件组。在继续阅读本章之前，你需要按下表定义组件与文件组之间

的关系。表 16.1 给出了本例所用的设置：

表 16.1 Sngl_Doc 安装程序的组件设置

组件	属性	取值
Program Files	Description	All the files required to run the application.
	Status Text	Copying Program Files ...
	Overwrite	SAMEORNEWERVERSION/ SAMEORNEWERDATE
	Required Components	Shared DLLs
	Included File Groups	Program Executable Files
Help Files	Description	Files that show you how to use the program.
	Status Text	Copying Help Files...
	Overwrite	SAMEORNEWERDATE
	Included File Groups	Help Files
Shared DLLs	Description	Common files used by the program.
	Status Text	Copying Shared DLLs...
	Overwrite	SAMEORNEWERVERSION/ SAMEORNEWERDATE
	Destination	<WINSYSDIR>
	Included File Groups	Shared DLLs

设置文件组

建立一系列的组件对安装程序并没有多大的帮助，它只是告诉用户哪些组件可以被复制到硬盘上，但还没有告诉安装程序要复制哪些文件，现在我们就来做这件事——为前面定义的文件组定义一系列文件。

先单击 File Groups 选项卡，你会看到带有所有文件组清单的 File Groups 对话框，如图 16.3 所示。选择要配置的文件组很容易，单击窗口左边给出的文件组列表项。

单击 Help Files 文件夹旁边的加号（+）标记，再单击 Links 列表项，你会看到如下图所示的 File Groups-Help Files\Links 对话框，从这个对话框中定义该文件组包含的一个或多个文件。

右击 File Groups-Help Files\Links 对话框，从上下文相关菜单中选择 Insert Files，你会看到标准的“打开文件”对话框，找到第 15 章建立的 Sngl_Doc.HLP 文件，单击 OK，就将它加入了 Help Files 文件组中。还需要把 Sngl_Doc.EXE 文件加入到 Program Excutable Files 文件组，最后一个文件组是 Shared DLLs，需要加入下列文件：MFC42.DLL 和 MSVCRT.DLL，它们将放在 Windows SYSTEM 文件夹下。


File Groups - Help Files\Links
Name
Size
Type
Modified
Link To

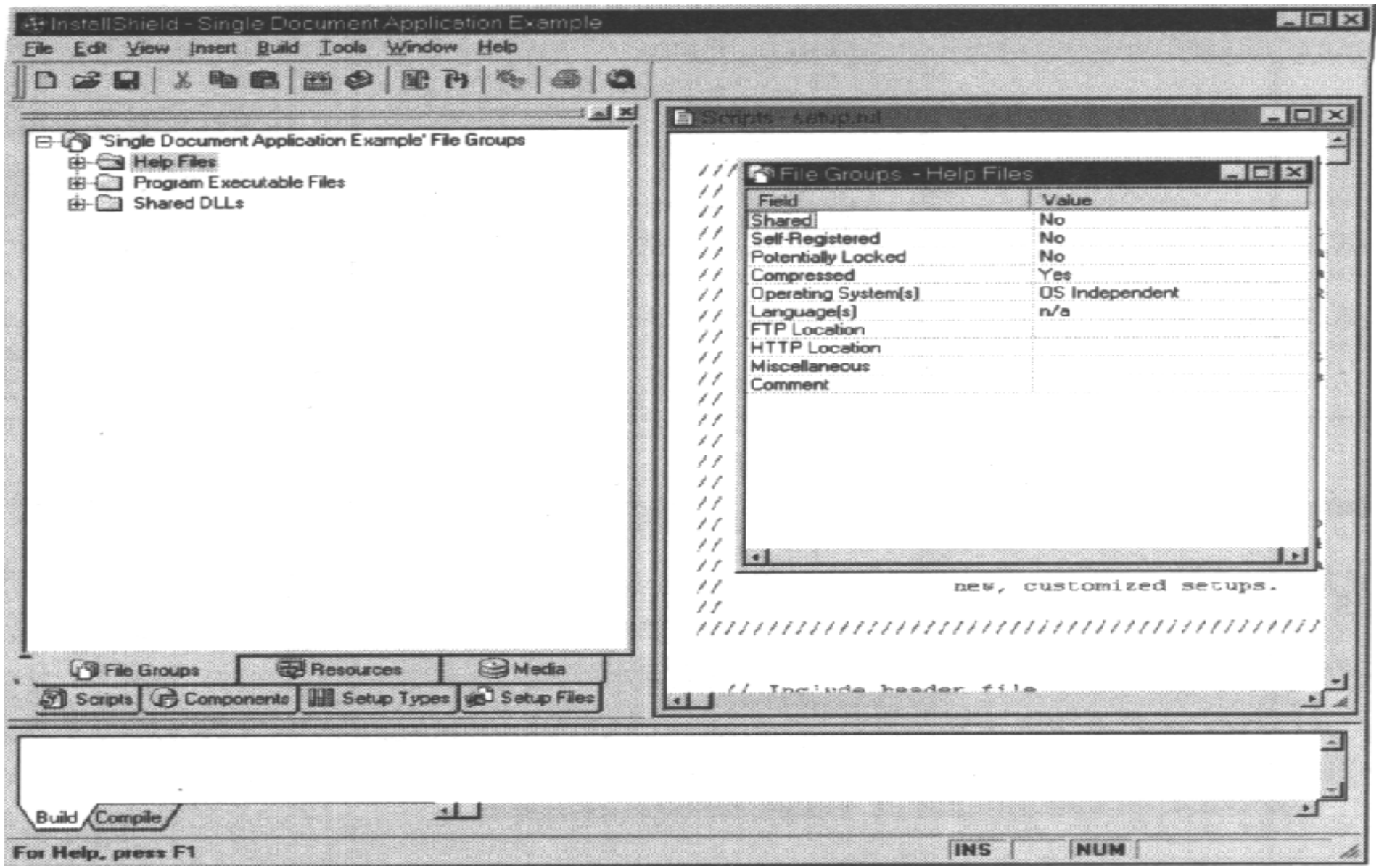


图 16.3　使用 File Groups - Help Files 对话框配置前面定义过的每个文件组

定义资源

到此为止，我们已经设置了组件和文件组。下面要给用户提供一些标识信息，因为他们当然希望对所安装的程序有所了解。选取 Resources 选项卡，你会看到如图 16.4 所示的 Resources 窗口，其中含有两个对话框：Resources - String Table 和 Resources - String Table\English。我们要在 Resources - String Table\English 对话框中进行修改。

在 Resources - String Table\English 对话框中进行的绝大多数修改并不影响安装程序，修改的目的是为了帮助用户理解安装程序进行的工作。改变资源值的做法和其它对话框改变属性的做法基本相同，只是双击要改变的 Identifier（标识符），InstallShield 就会显示相应的对话框供修改，下图是修改 COMPANY_NAME 标识符的示例：

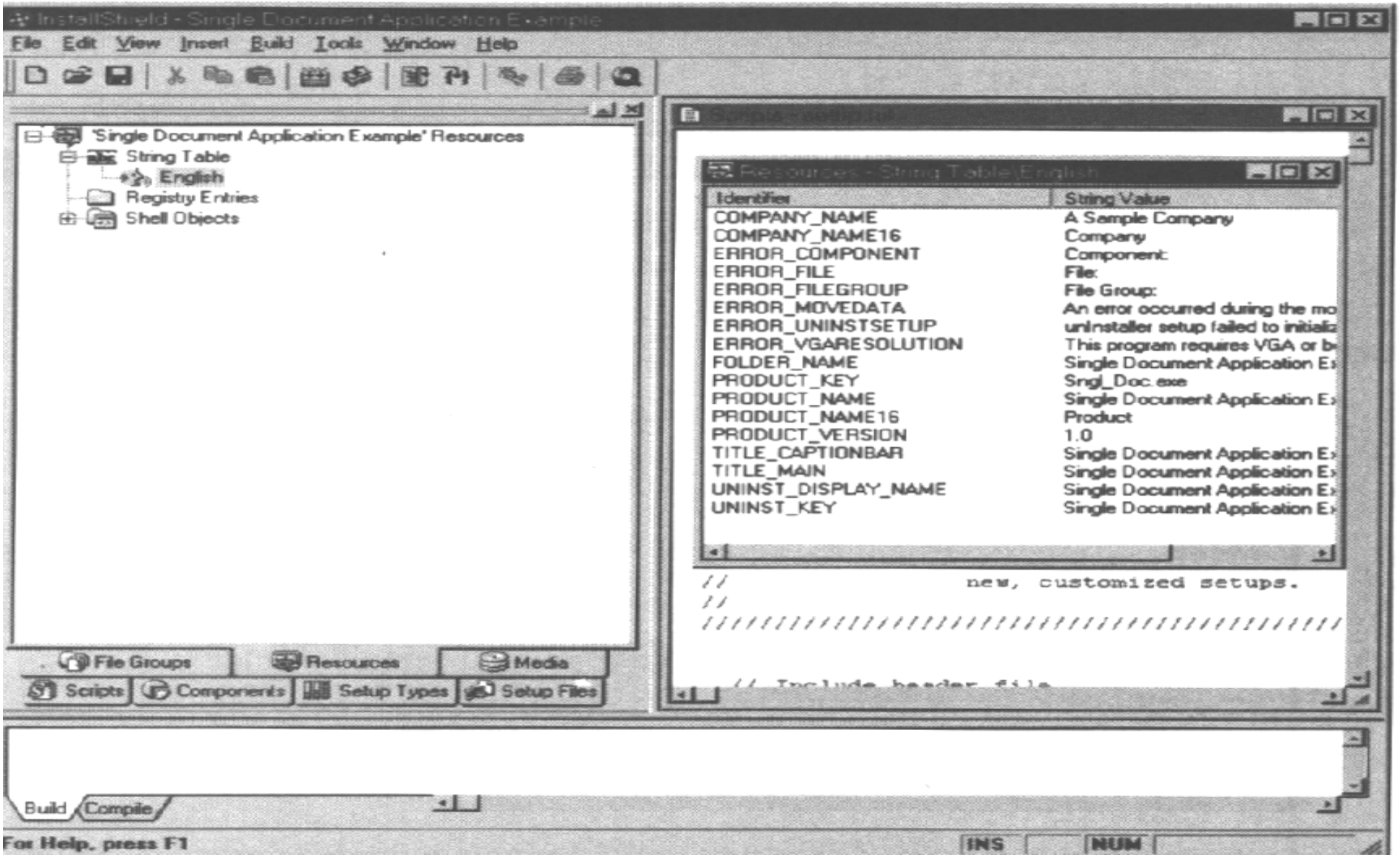


图 16.4　InstallShield 几乎与 Visual C++相同的方式使用资源

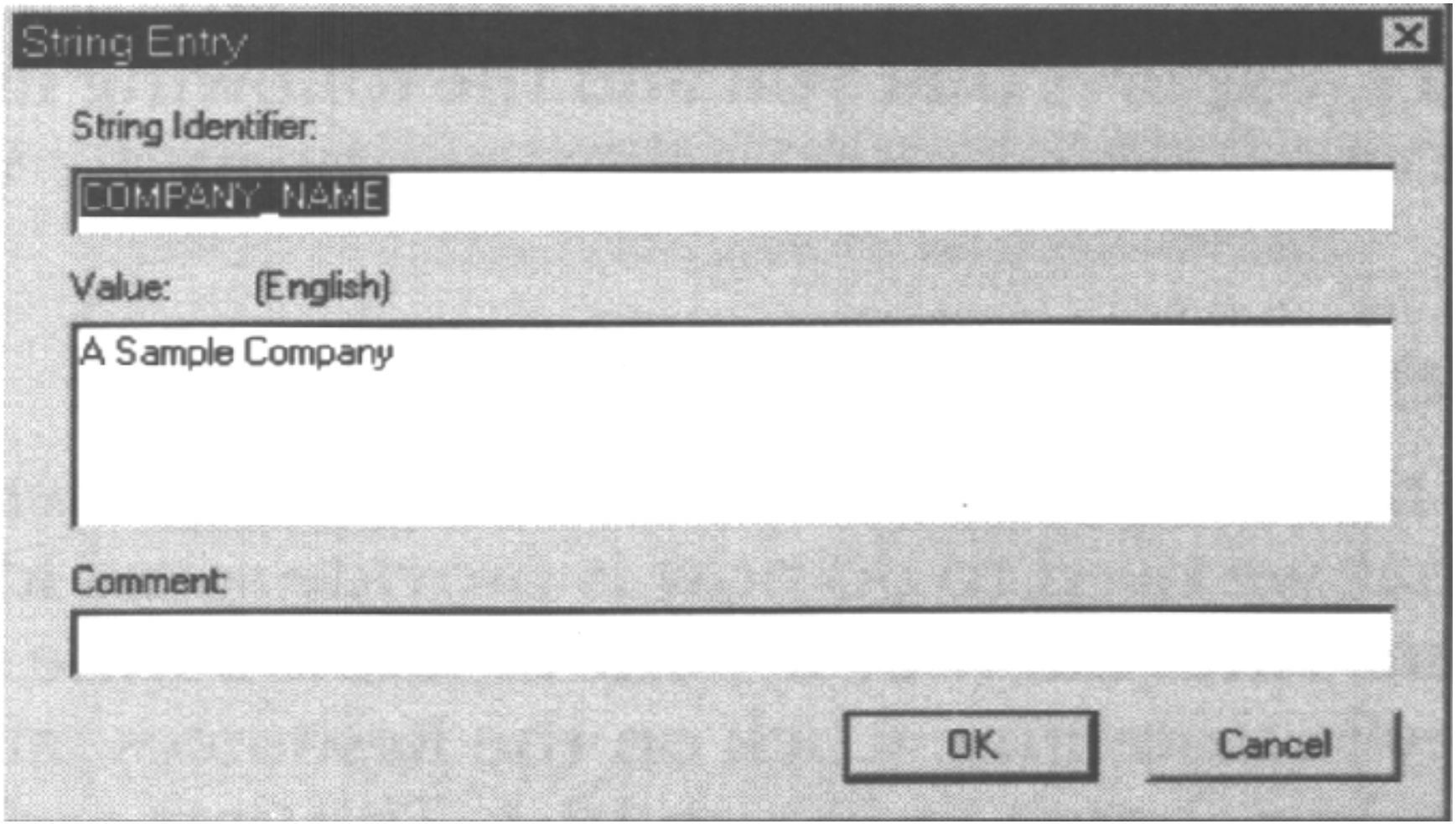


表 16.2 是我们对示例程序所做的修改——显然，各标识符的取值依据安装程序的不同而不同。

表 16.2 Sngl_Doc 安装程序的字符串值

标识符	取值
COMPANY_NAME	A Sample Company
PRODUCT_KEY	Sngl_Doc.EXE
PRODUCT_NAME	Single Document Application Example
UNINST_DISPLAY_NAME	Single Document Application Example
UNINST_KEY	Sngl_Doc.EXE

决定安装介质类型

现在我们要做配置安装程序必须完成的最后一件事了：就是告诉 InstallShield 要使用何种介质发行你的应用程序。在绝大多数情况下，你会使用 CD 作为主要的发行介质，如果应用程序很小的话，也可以使用软盘作为另一种发行方法。如果你只是选择 CD 作为唯一的发行介质，那么现在你不用再做什么了。否则的话，选择 Media 选项卡，你会看到如图 16.5 所示的 Media 对话框及窗口：

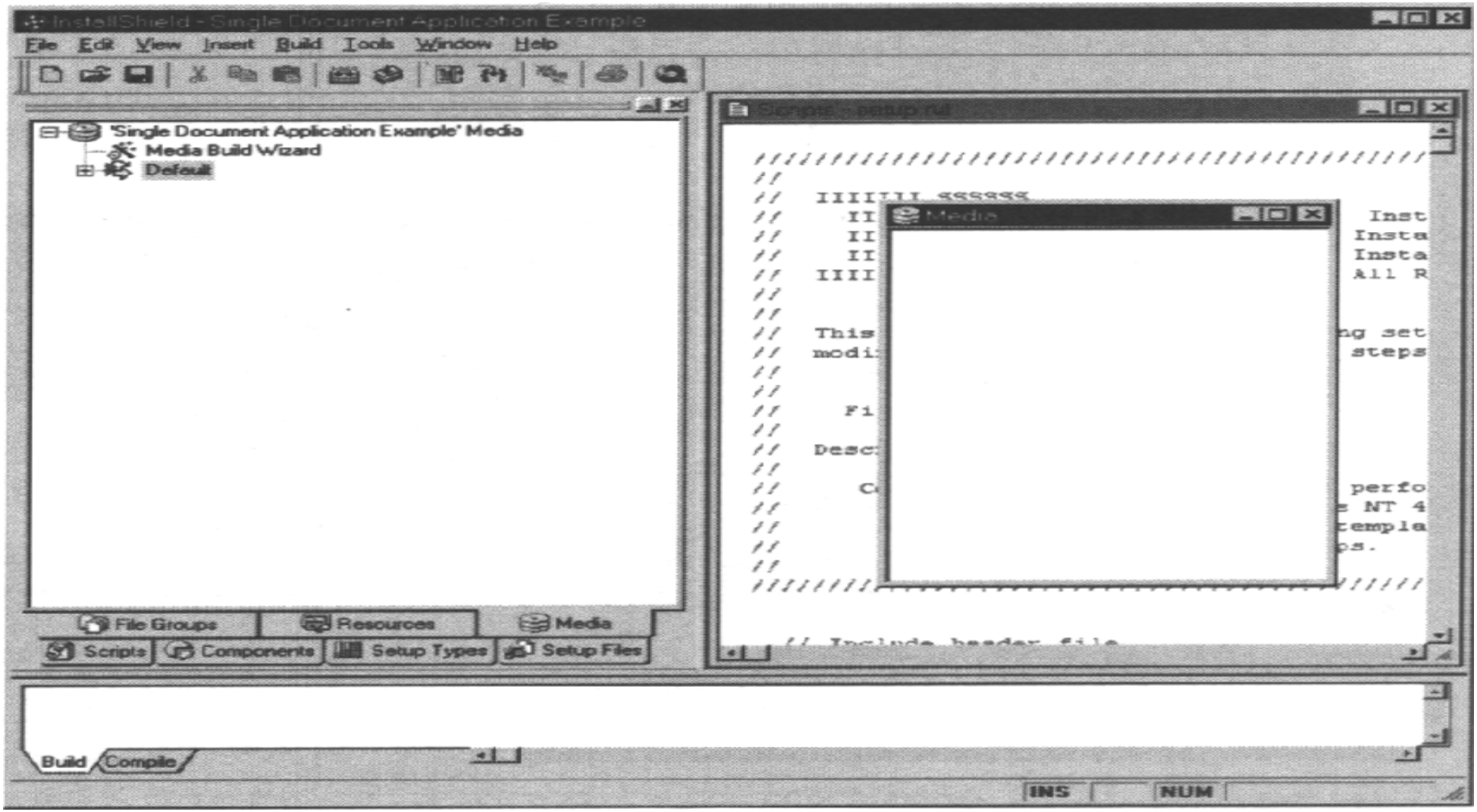
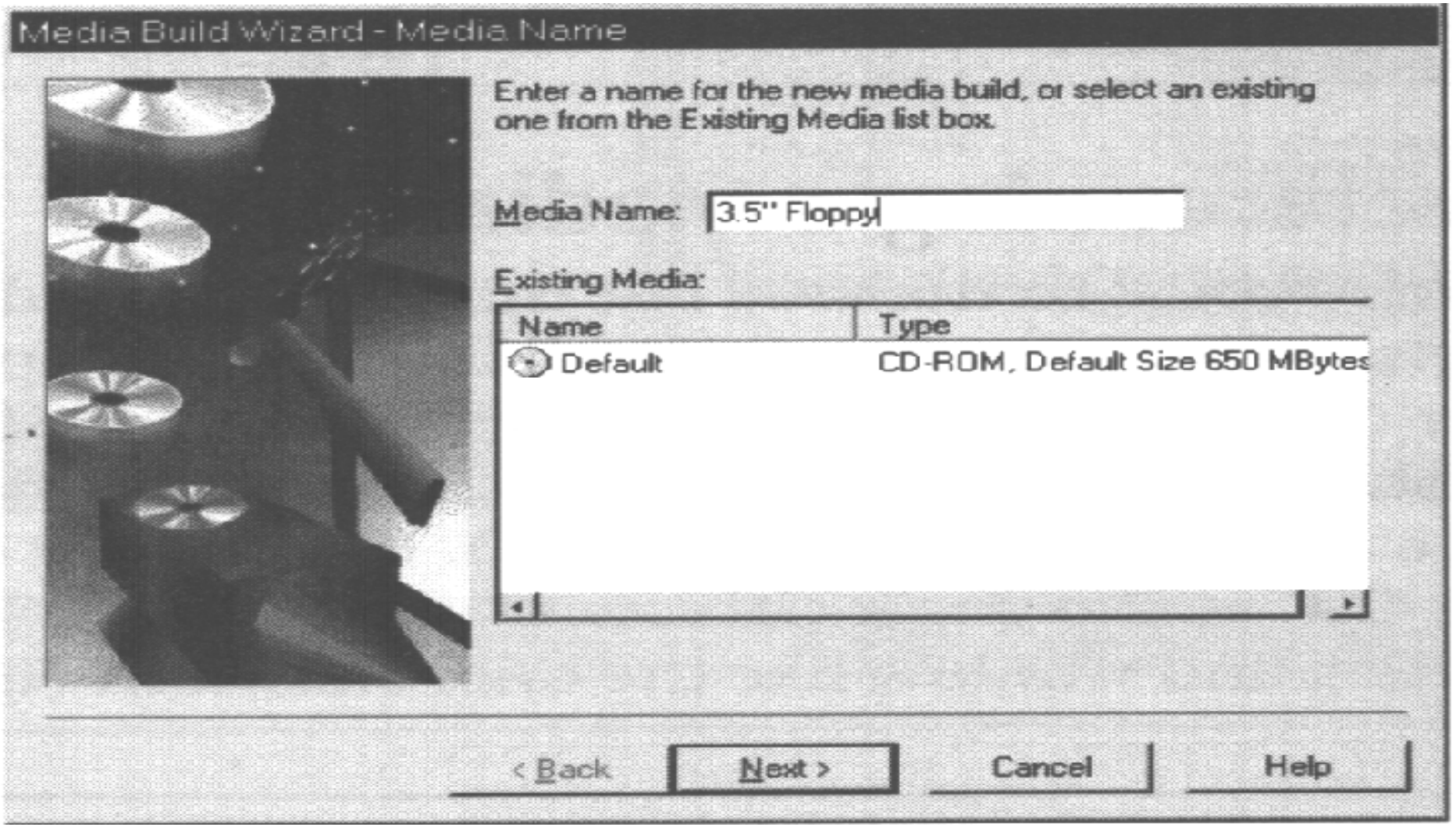


图 16.5　用 Media 选项卡可以定义除 CD 以外的发布介质

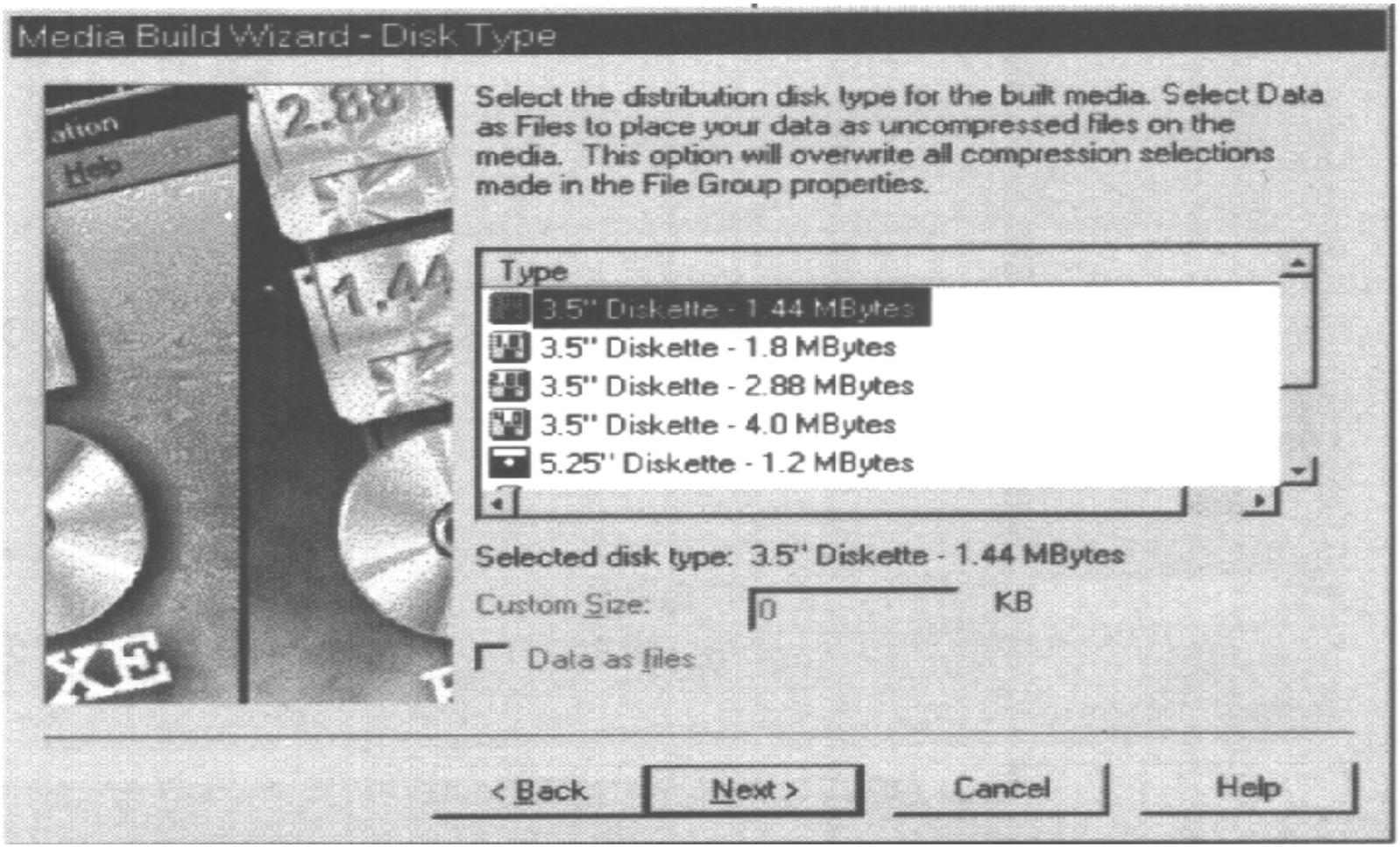
正如你所看到的，CD 已经定义好了（如图 16.5 的 CD 文件夹所示）。需要用 Media Build Wizard 向安装程序中加入其它的介质类型，下面的步骤用于

增加软盘介质类型：

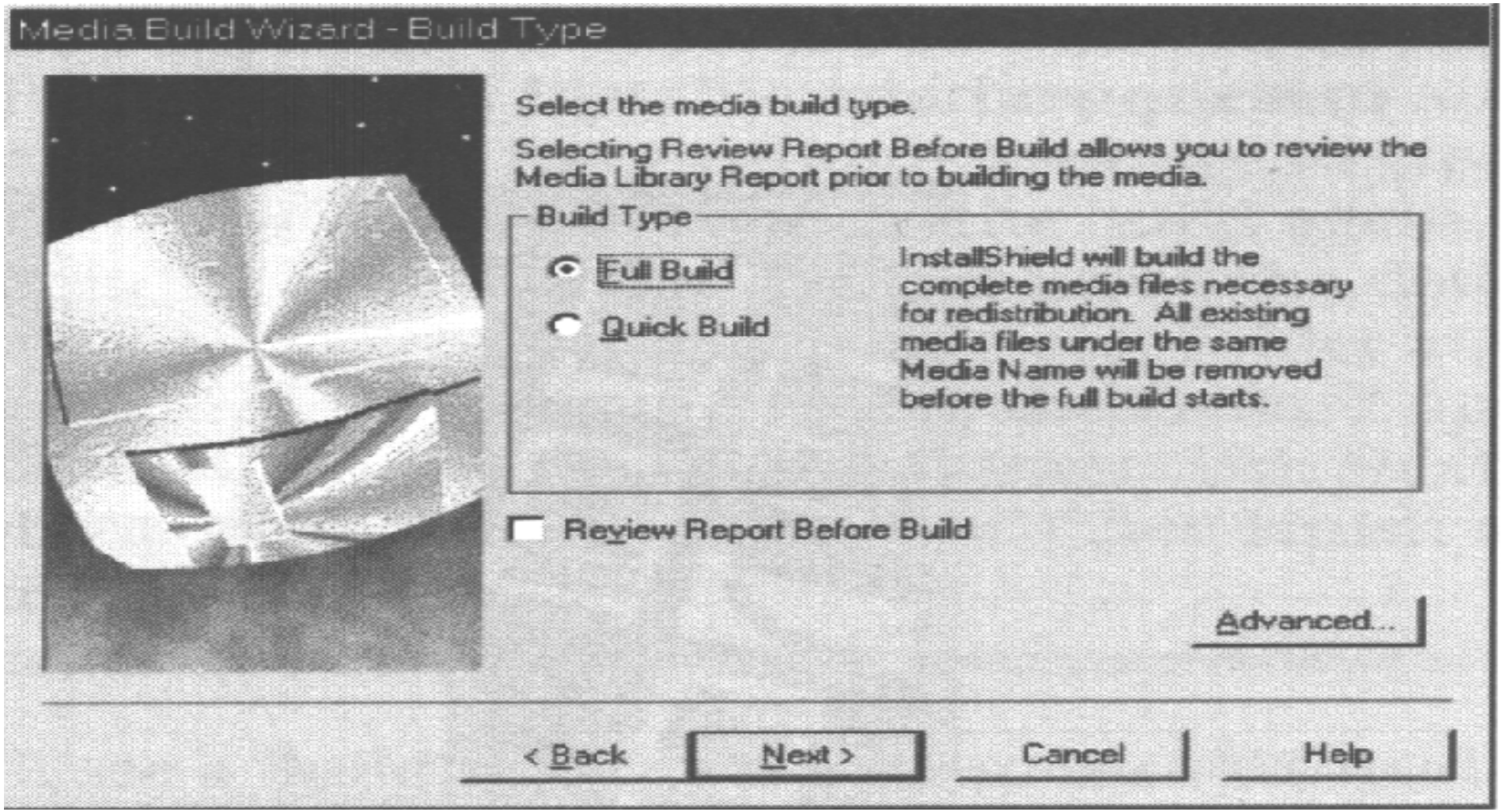
1. 单击 Media Build Wizard 选项，你会看到 Media Build Wizard - Media Name 对话框，如下图所示：



2. 输入 3.5" Floppy 后，单击 Next，你会看到如下图所示的 Media Build Wizard - Disk Type 对话框。请注意，InstallShield 支持各种各样的介质类型，其中包括 2.88MB 软盘。

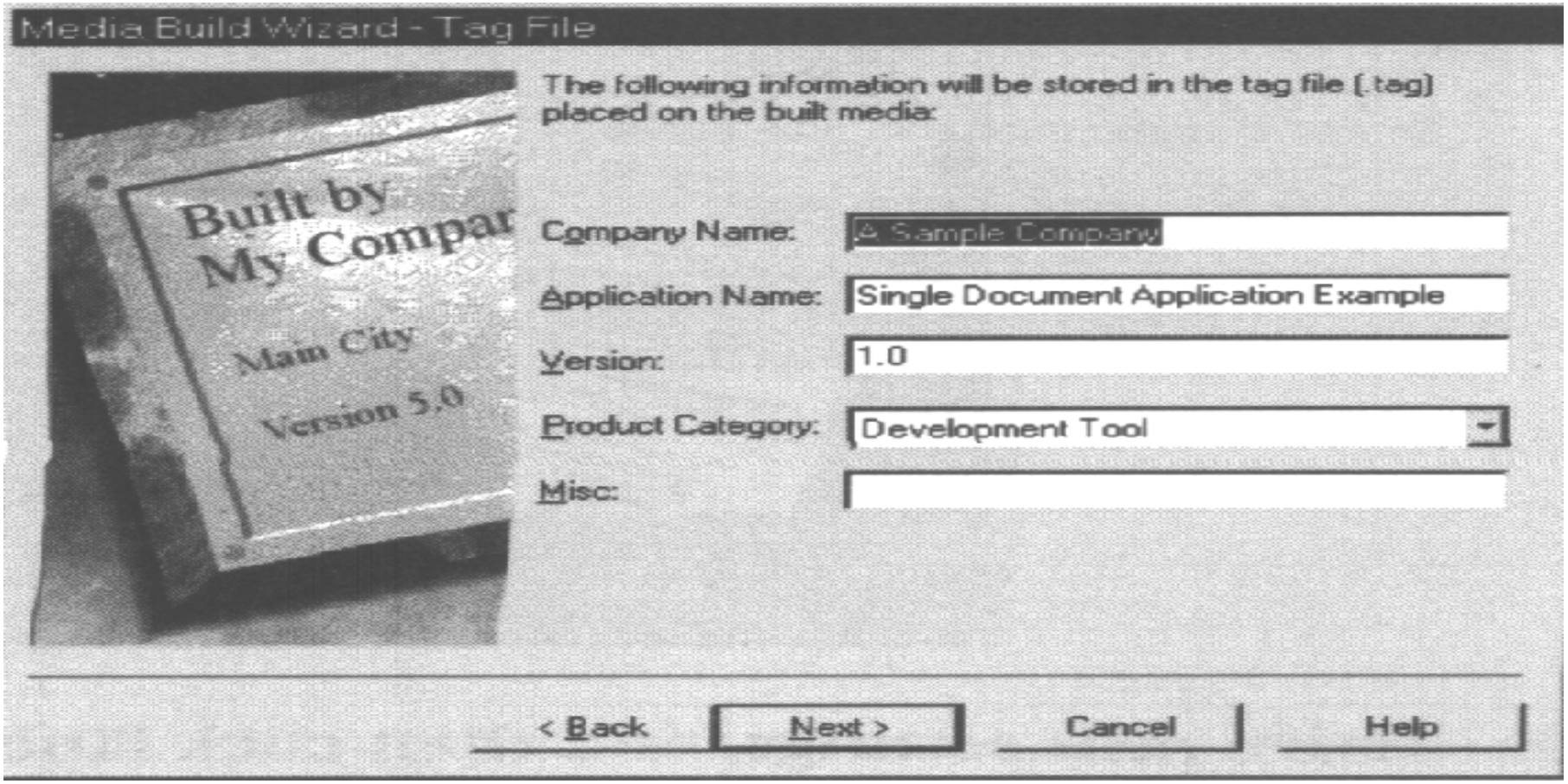


3. 选取 3.5" Diskette-1.44MByte 选项，然后单击 Next，你会看到如下图所示的 Media Build Wizard - Build Type 对话框：



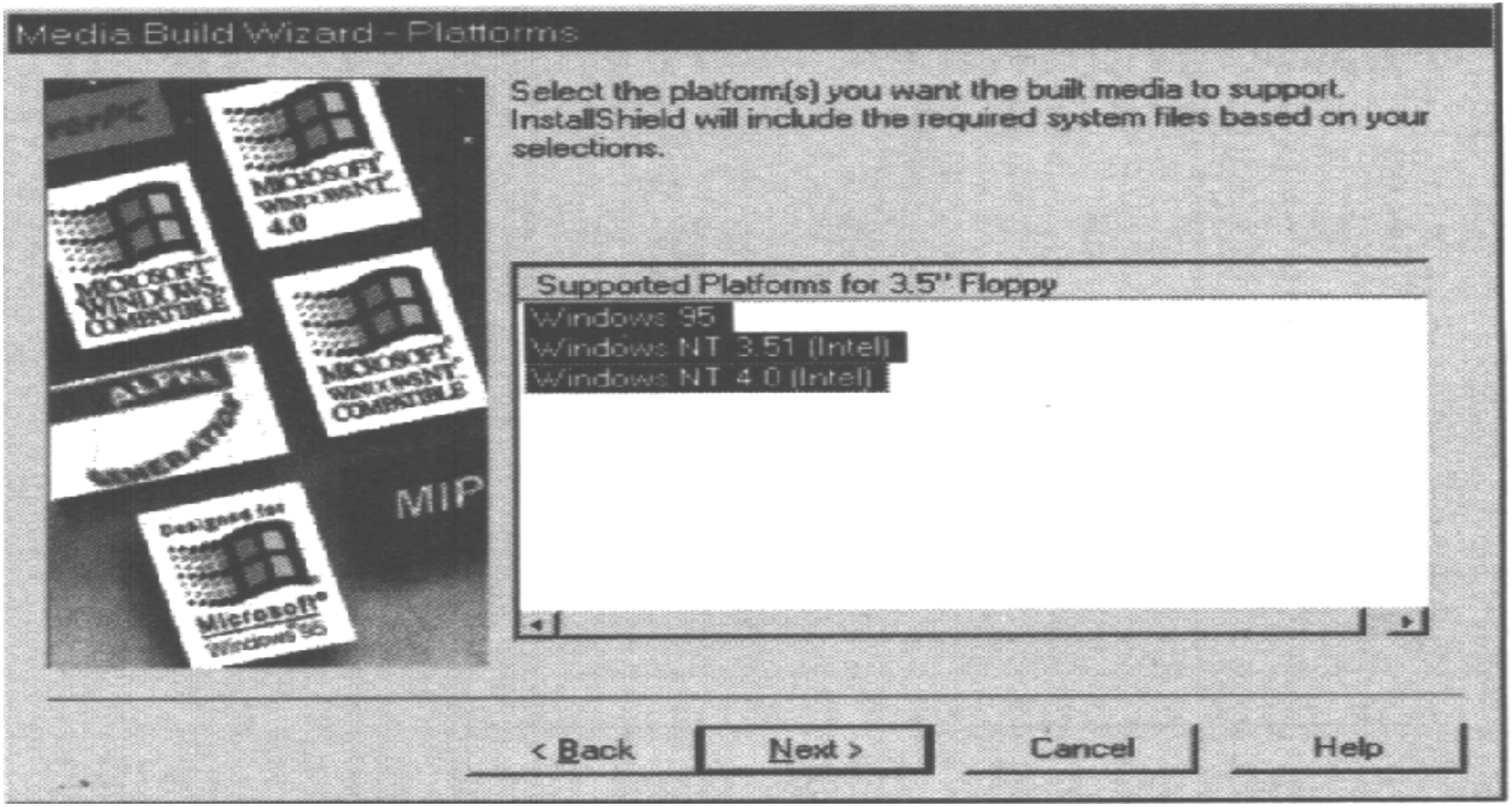
选择 Full Build 选项时，系统将压缩应用程序中包含的所有文件、建立 CAB 文件并创建全功能的安装程序。Quick Build 选项用于测试安装程序是否正确。

4. 选择 Full Build 选项，然后单击 Next，你会看到如下图所示的 Media Build Wizard - Tag File 对话框，从这里输入公司名称及相关的应用程序信息。

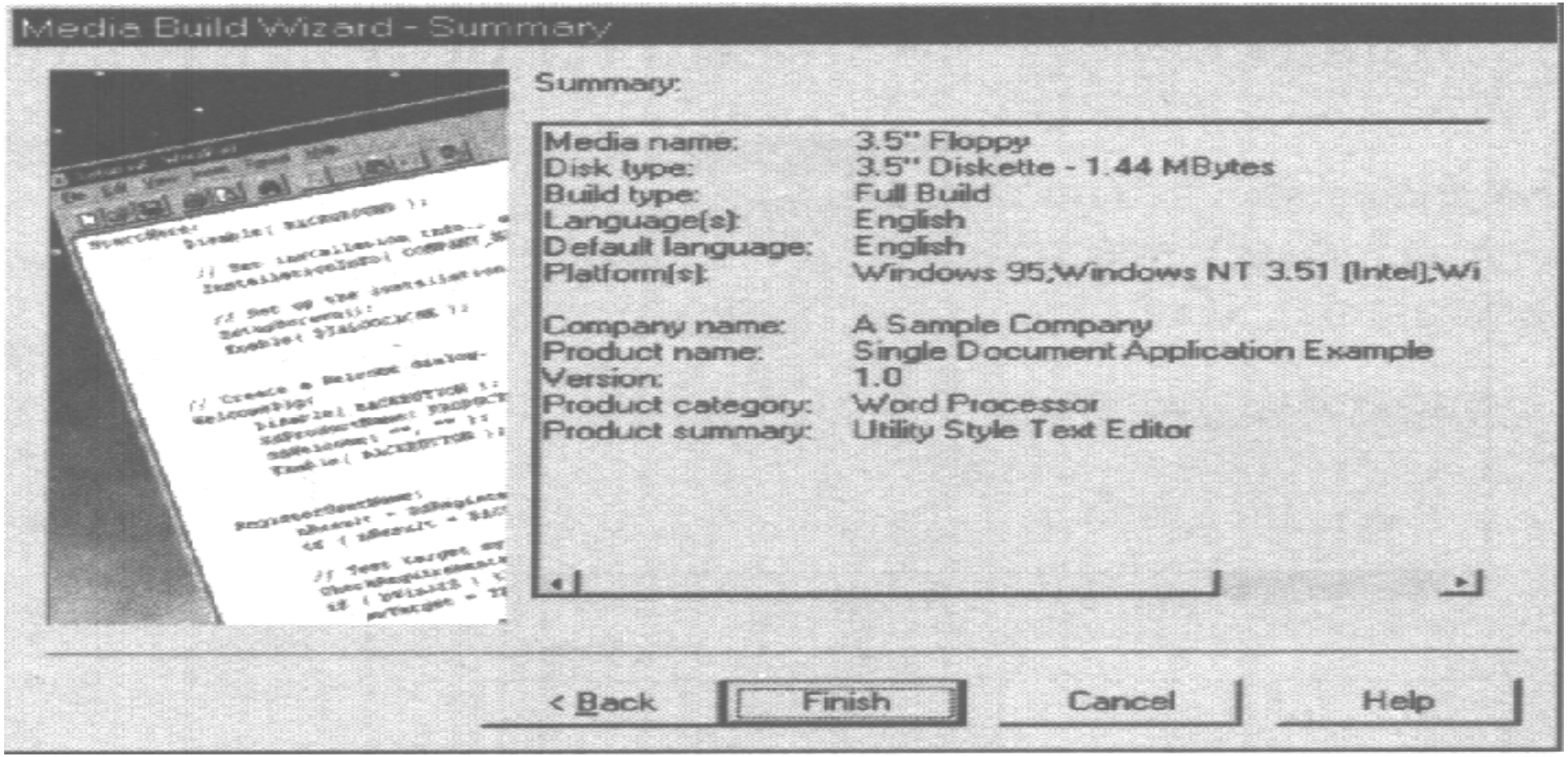


5. 在这个对话框中输入所有需要的信息。示例程序在 Company Name 域使用的是 ABC Corporation，在 Application Name 域使用的是 Single Document Application Example，在 Product Category 域使用的是 Word Processor，在 Misc. 域使用的是 Utility Style Text Editor。

6. 单击 Next，你会看到如下图所示的 Media Build Wizard - Platforms 对话框：



7. 单击 Next，你会看到如下图所示的 Media Build Wizard Summary 对话框，这是验证所有设置正确性的最后机会。



8. 检查所有设置，正确无误后单击 Finish，InstallShield 创建新的、你所需要的安装程序。

注释 InstallShield 的 Free Edition 版此时会显示一条错误信息，指示没有足够的内存来创建安装程序。可以忽略这个问题，然后重试，如果还不正确，使用缺省的安装设置进行创建。做法是：右击 Default

文件夹，从上下文相关菜单中选择 Media Build Wizard，在 Media Build Wizard 对话框的第一个显示中选择 Default 选项，然后单击 Next，直到不再出现错误消息，按 Finish 按钮，InstallShield 创建所需介质类型。

这时你会看到 Building Media 对话框，InstallShield 自动创建安装程序，安装程序创建后，Building Media 对话框如下图所示，只需单击 Finish 按钮，这下就全部完工了！