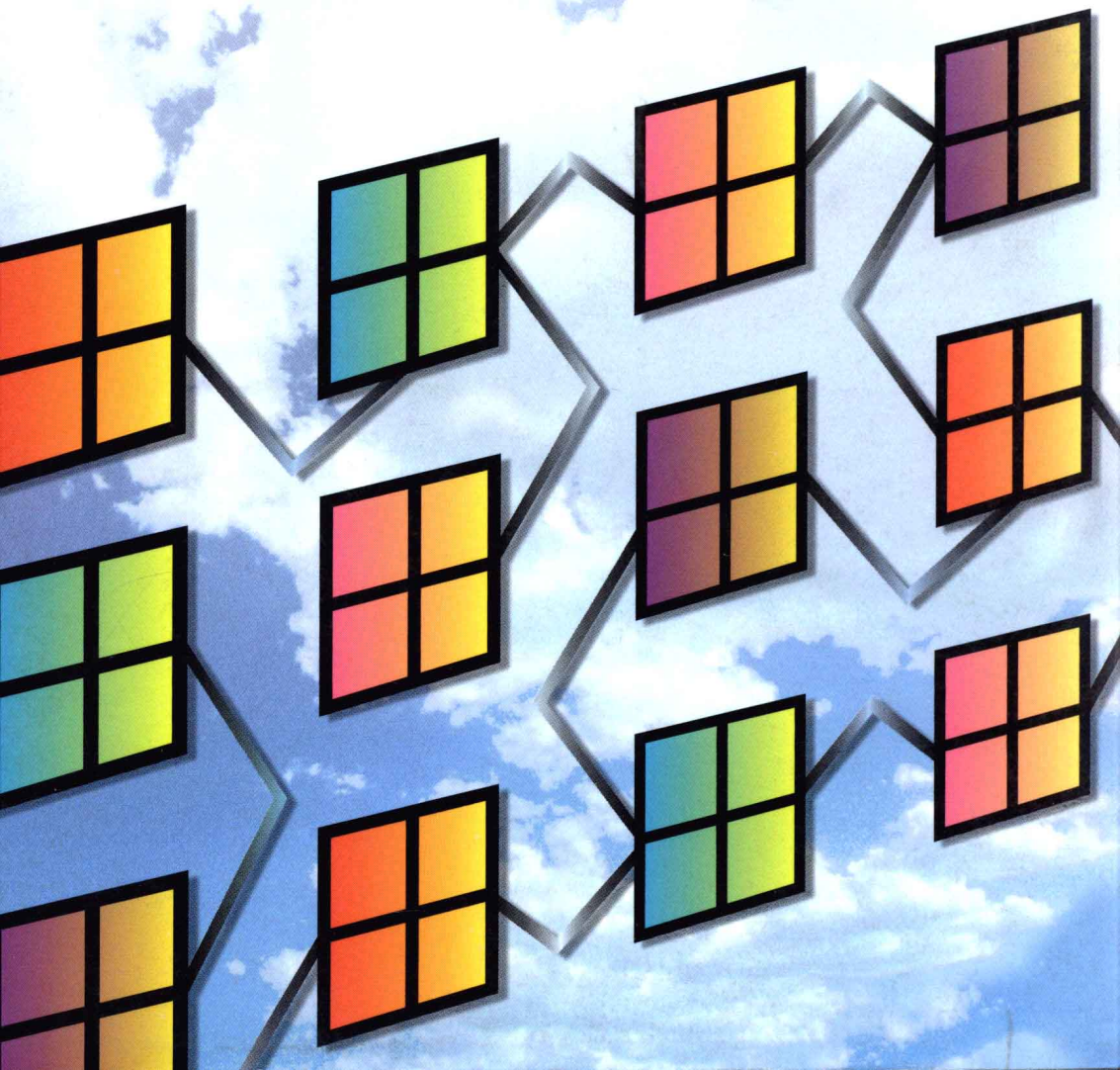
Covers Windows for Work Groups

Network Programming in Windows NT™

Alok K. Sinha



Network Programming in Windows NT™

Alok K. Sinha
Microsoft Corporation

ADDISON-WESLEY PUBLISHING COMPANY

Reading, Massachusetts • Menlo Park, California • New York
Don Mills, Ontario • Wokingham, England • Amsterdam • Bonn
Sydney • Singapore • Tokyo • Madrid • San Juan • Milan • Paris

This book is in the **Addison-Wesley UNIX and Open Systems Series.**
Series Editors: Marshall Kirk McKusick and John S. Quarterman

Senior Acquisitions Editor: Thomas E. Stone
Associate Editor: Deborah Lafferty
Savannah Consultants Editor: Bonnie Mae Savage
Production Supervisor: Nancy H. Fenton
Cover Designer: Barbara Atkinson
Production: Editorial Services of New England
Project Manager: Bonnie Jo Collins
Text Designers: Pat Nieshoff/Nieshoff Design, Mark Heffernan
Text Figures: George Nichols
Copy Editors: Sandra Sizer Moore, Beverly Miller
Senior Manufacturing Manager: Roy Logan

Access the latest information about Addison-Wesley books from our Internet gopher site or visit our World Wide Web page:
`gopher aw.com`
`http://www.aw.com/cseng/`

Library of Congress Cataloging-in-Publication Data
Sinha, Alok.
Network programming in Windows NT / Alok Sinha.
p. cm.
Includes bibliographical references and index.
ISBN 0-201-59056-5
1. Microsoft Windows NT. 2. Computer networks. I. Title.
QA76.76.063S567 1996
005.2—dc20 95-31136
CIP

Printed in the United States of America

1 2 3 4 5 6 7 8 9 10—MA—99 98 97 96 95

Dedication

To Bonnie Mae Savage

Preface

I was not always interested in networking, or computer science for that matter. In fact, I received a master of science degree in mining engineering from University of Alaska, Fairbanks. After graduating, I was looking for something interesting to do while my would-be wife completed her graduate degree in civil engineering, so I signed up to get a graduate degree in computer science. The first year's courses were full of complexity theory and algorithms, both of which I enjoyed but they didn't thrill me. In the second year, I took the Networking Architecture class from an unorthodox professor, Dr. Peter Knoke, who had recently come to academia after decades of industry work. Instead of following the prescribed book, he narrated the latest technology developments straight out of trade magazines and newspapers (the *Wall Street Journal* was his favorite). Our individual projects involved research in the latest areas of networking, summarized in in-depth presentations. I picked local area networks as my topic and scoured all the trade magazines for information. When I hit an unknown term, such as "router," I would pore over the books and research papers to understand the technology and architecture. By the end of the semester, I had a broad knowledge not only of the theory of networking but also of the latest technology. Needless to add, I was hooked on networking and have spent most of my professional career in and around networking—from Microsoft LAN Manager to Directory Service to interactive TV applications. Most important, however, Dr. Knoke succeeded in showing his students how to learn on their own in a rapidly changing industry—a vital skill for surviving in the computer industry.

The Audience

This book is targeted toward software developers and project managers who want to learn about using the interprocess communication (IPC) mechanisms offered by Windows and Windows NT for creating client/server applications. Actually, students will also find this book useful in developing a new generation of networking applications in their laboratories. I assume that the reader is familiar with Microsoft Windows and Windows NT programming, has working knowledge of C programming, and understands the basics of networking.

The first three chapters of the book present the background for network programming in Windows and Windows NT environments. These chapters discuss the networking architecture of Windows for Workgroups™ and Windows NT as well as certain advanced programming features such as Win32 services. The other six chapters explain the concepts and use of these IPC mechanisms in Windows and Window NT environments: Remote Procedure Call (RPC), Windows Sockets, Named Pipe, Mailslot, Network Basic Input/Output Specification (NetBIOS), Internet Packet Exchange™/ Sequenced Packet Exchange™ (IPX/SPX).

Within Chapters 4 through 9, I first present the basic concepts and a primary-use model of an IPC mechanism in the Windows NT environment and then proceed to convey the details of using the given IPC mechanism by explaining the necessary system calls, showing sample programs illustrating programming details, and unfolding intricate details wherever pertinent. I then present any special programming details necessary to use the IPC mechanism from an application in the Windows environment.

Network programming is like any other discipline of computer science wherein one optimizes every line of code, tweaks every algorithm, to maximize efficiency and performance. But it differs from other disciplines in that a significant amount of effort needs to be spent on anticipating the inevitable network and software failures. Such upfront concern and understanding allow you to build software so robust that your applications continue to function despite such failures.

Acknowledgments

This book takes its current form due to the constructive critiques of Bonnie Mae Savage, the local editor and program manager at Savannah in Redmond, Washington, who shaped the focus and the content of the book. Also formative were the constructive critiques of the reviewers and the editorial talents at Editorial Services of New England. Together, they have all molded a raw technical blurb into a coherent, extremely readable, and I hope a downright enjoyable networking book. I thank their persistence, patience, tenacity, and every other human virtue I forgot to mention! In addition, I must thank Tom Stone at Addison-Wesley for accepting my half-page e-mail as the table of contents for almost half a year! I also thank all individuals who have directly and indirectly helped this book: Deborah Lafferty from Addison-Wesley for her patience and good counsel; Roger Soles from Oracle Corporation; and James "J" Allard, Manny Weiser, David Treadwell, Ryszard Kott, Margaret Johnson, and Jim Allchin from Microsoft Corporation.

Alok Sinha
August 1995

Contents

Chapter One

INTRODUCTION

Microsoft® Windows™ operating system is the predominant **personal computer (PC)** operating system, and one of the most popular desktop environments in use today. Windows users can run their productivity tools, such as word processors and spreadsheets, in a consistent, simple graphical user environment. However, the original Windows system can only be used on PCs containing Intel® x86-based processors. Further, Windows is not suitable for computing-intensive applications, such as **computer-aided design (CAD)** applications, nor is it robust enough to handle "mission-critical" applications such as accounting and payroll. Although such applications are not appropriate for the original Windows operating system, they can be hosted on the Microsoft Windows NT™ operating system.

Whether they are running the Windows or Windows NT desktop operating system, networked users need to access file servers to store information and print servers to print documents. To do either of these tasks, most users connect their PCs to a **local area network (LAN)**. Today's LAN-based file servers and print servers provide access to information storage and printers, and also to many other networked resources. There is a growing demand for desktop access to such new services as database servers, client/server computing, and the Internet. For these reasons, software developers are now focusing on a new breed of applications. In this next level of computing, applications running on the desktop interact with other applications that might reside on one or more remote machines. This defines network computing, in which applications on one machine communicate with their counterparts on a remote machine using an **interprocess communication (IPC)** method.

This book, *Network Programming in Windows NT*, introduces software developers, network administrators, and others to the methods of building client/server systems on Windows and Windows NT IPC interfaces. It also provides insights into the network architectures of Windows and Windows NT. By understanding the basic architectures of these environments and by following the explanations of the IPC mechanisms, you should be able to design and implement efficient Windows and Windows NT applications that can harness the power of network computing.

Network Programming is intended primarily for people in the software development community who are (or who plan to be) designing client/server software for Windows and Windows NT environments: software developers writing new applications for Windows and Windows NT environments; engineers porting applications from UNIX® and OS/2™ environments to the Windows NT environment; project managers involved in developing Windows and Windows NT-based applications; software consultants; and students who are using Windows and Windows NT environments to learn the principles of networking. To benefit the most from this book, you should be familiar with Windows and Windows NT programming, have a working knowledge of the C programming language, and a basic understanding of networking. Some advanced sections of the book may be easier to grasp if you are familiar with the concepts of object-oriented programming. All sample programs are written in C and have been compiled in Microsoft Visual C™ 2.0 environment for Windows and Windows NT. All sample programs can be downloaded from **`ftp://ftp.aw.com/cseng/sinha/windowsnt`** via anonymous **file transfer protocol (FTP)** on the Internet.

The book is organized to help you understand each IPC mechanism quickly. The first three chapters present the background for network programming in Windows and Windows NT environments. They cover the networking architecture of **Windows for Workgroups™ (WFW)** and Windows NT and certain advanced programming features. Readers already familiar with these concepts may want to skip these chapters or use them as a reference. However, the remaining chapters assume the knowledge of these materials.

Chapters 4 through 10 explain the concepts and use of IPC mechanisms in Windows and Windows NT environments: **remote procedure call (RPC)**, Windows Sockets, Named Pipe, Mailslot, **Network Basic Input/Output Specification (NetBIOS)**, and **Internet Packet Exchange™/Sequenced Packet Exchange™ (IPX/SPX)**. Each chapter begins with the basic concepts and a primary-use model of the IPC mechanism in the Windows NT environment. It examines the details of using the IPC mechanism, explains the necessary system calls, shows sample programs illustrating programming details, and unfolds intricate details wherever pertinent. It presents any special programming details necessary for using the mechanism in an application in the Windows environment and ends with a summary of the IPC mechanism's highlights.

Each chapter discusses issues that can affect the reliability or the performance[1] of an application when you use a feature, and suggests ways you can improve reliability and performance. Whenever pertinent, I present comparisons

1. Performance of a network-aware application is often measured in terms of a number of matrices, such as the time delay involved in network I/O, the resource overhead per user, the total data throughput per second, and so on.

among features of the different IPC mechanisms, but I do not compare the performance of one IPC mechanism with another in Windows or Windows NT environments. (Given the wide range of possible test scenarios, performance matrices, and software upgrade packages for Windows and Windows NT, performance comparisons between one IPC with another are outside the scope of this book.) I also briefly address Windows NT security concepts, which can be used to design secure software systems; security is an important consideration in any network-aware software system.

Throughout this book, the term "communication methods" refers to IPC over a network by some physical medium such as Ethernet. (For the purposes here, other communication methods, such as serial communication by **communications (COM)** ports and modems, are irrelevant.) Whenever specific differentiation is not necessary, the versions of MS-DOS-based Windows (Windows 3.0,™ Windows 3.1,™ Windows 3.11,™ and Windows for Workgroups 3.11,™ etc.) are referred to as Windows. Similarly, unless explicitly stated otherwise, Windows NT refers to all versions on Windows NT platforms (Windows NT 3.1, Windows NT 3.5 Workstation, Windows NT 3.5 Advanced Server, and so on). The evolution of the Windows and Windows NT operating systems is illustrated in Figure 1-1. Finally, the term Windows NT Workstation is used except when the distinction between Windows NT Workstation and **Windows NT Advanced Server (NTAS)** products is important for the discussion at hand.

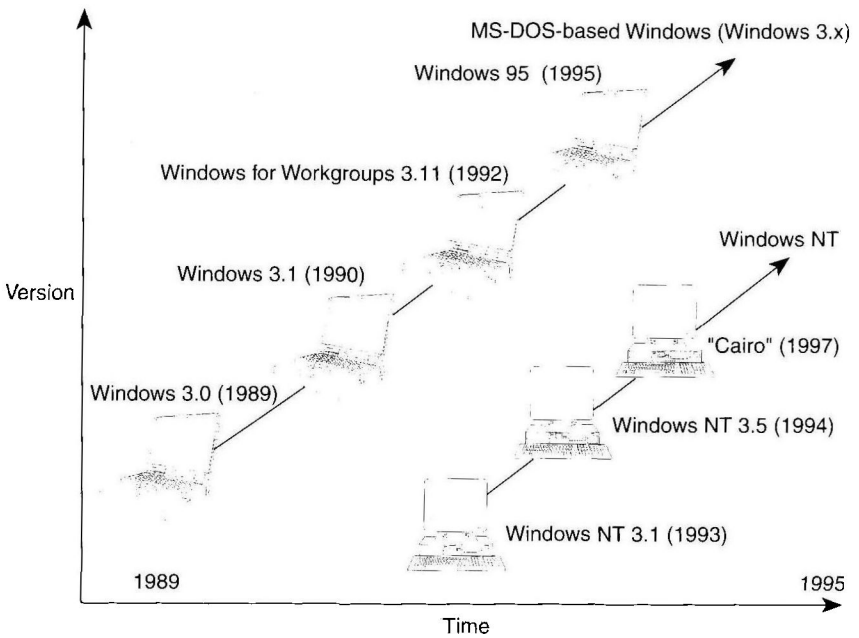


Figure 1-1. *Evolution of Microsoft Windows and Windows NT Operating Systems*

1.1 HISTORICAL PERSPECTIVE

The arrival of the Windows 3.0 operating system in 1989 caused a major shift in the way people used IBM®-PC compatibles, a shift no less important than the introduction of the PC in the early 1980s. Before the widespread use of Windows, PCs were operated by a simple **Microsoft disk operating system (MS-DOS®)**, consisting of a command shell with a series of nonintuitive commands and utilities. Most users found MS-DOS unwieldy and nonfriendly. Furthermore, the MS-DOS environment did not specify any user interface guidelines. Thus, each MS-DOS–based application had its own user interface, which usually bore little resemblance to other applications. Today, most software houses and **independent software vendors (ISVs)** are delivering their second or third generation of Windows-based applications. As you walk down the aisles of your neighborhood stores, you can easily spot rows of computers running Windows. What promoted such popular acceptance of Windows 3.0 was its simple, user-friendly **graphical user interface (GUI)** and consistent use of Windows **application programming interfaces (APIs)**.

The Windows operating system simplified and standardized the look and feel of all Windows-based applications. Although it was not always stable, the Windows 3.0 operating environment was complete enough that users did not have to face the MS-DOS prompt ever again—unless they wanted to. Windows, in establishing a consistent look and feel for application GUI and by standardizing APIs, made it possible for users to adapt quickly to Windows-based programs. This shift should not surprise anyone, given the success of Apple Macintosh® (starting in 1984) and the proven work on user interfaces done at Xerox® Labs by Alan Kay and his group. GUIs just took time to arrive in the PC market because of the sheer lack of standards, the shortage of sophisticated software and hardware, and the inability of the major players to forge agreements.

Another reason for the success of Windows has been the immense number of application programming interfaces developed for it, about 700 in Windows 3.0 and about 1,100 in Windows 3.1. These standardized interfaces have allowed the applications to use common ways to access resources and system services. They have also reduced the time needed to develop applications.

Windows does have some limitations as an operating system. It is based on MS-DOS and allows only nonpreemptive multitasking. Thus, an errant application can refuse to relinquish control of the **central processing unit (CPU)** while starving other applications of computing cycles. This problem has been resolved in newer versions of Windows (Windows 95) as well as in Windows NT. Another issue was that networking capabilities were not tightly integrated within Windows 3.0. To solve networking woes with Windows, Microsoft offered Windows for Workgroups 3.11 in 1992. This package not only made networking easier, but it also allowed peer-to-peer communication, file sharing and printing, and mail among members of a workgroup.

In 1985, nearly parallel to the Windows evolution, Microsoft and IBM began developing a true multitasking operating system with a GUI and based on a new architecture, not MS-DOS. This work led to early versions of IBM OS/2.™ IBM's version did not attract a major following for many reasons. In 1989, rather than fixing OS/2, Microsoft started designing a new operating system. The new system is known today as Windows NT.[2] This multitasking operating system is designed to:

- Port to varying hardware, including the **reduced instruction set computing (RISC)** processors
- Scale to multiple processor systems
- Integrate with networking, thus allowing client/server computing capability
- Be POSIX[3] compliant
- Meet C2[4] government security certification

While Windows NT has a completely new architecture, its designers made some major decisions that empower users and application developers alike. First, Windows NT has the same user interface as that of MS-DOS–based Windows. Second, most existing Windows and MS-DOS applications can be executed on Windows NT regardless of the underlying hardware. For example, MS-DOS applications can run in Windows NT on a MIPS® RISC machine. In short, Windows NT is largely backward compatible.

Thus, Microsoft has provided two operating systems that incorporate similar user and application programming interfaces. Because Windows NT has been designed to be the state-of-the-art operating system that is also backward-compatible with MS-DOS–based Windows, users can run Windows on computers with limited resources, and they can use Windows NT on high-end computers as shown in Figure 1-2.

1.2 IMPACT OF WINDOWS NT DESIGN ON USERS

The Windows NT operating system, much like the UNIX operating system, can run on diverse hardware platforms. The user interface remains the same across all Windows systems; users can move easily from one system to another, and quickly learn and adapt to new applications. While businesses use Windows operating systems for running productivity applications, such as word processors and spreadsheet applications, the scientists, engineers, and

2. See Helen Custer's book *Inside Windows NT* for insight into the design of the NT operating system.
3. POSIX is the portable operating system interface based on UNIX, as defined by IEEE standard 1003.1-1988.
4. See Department of Defense Trusted Computer System Evaluation Criteria, DOD 5200.28-STD, for details.

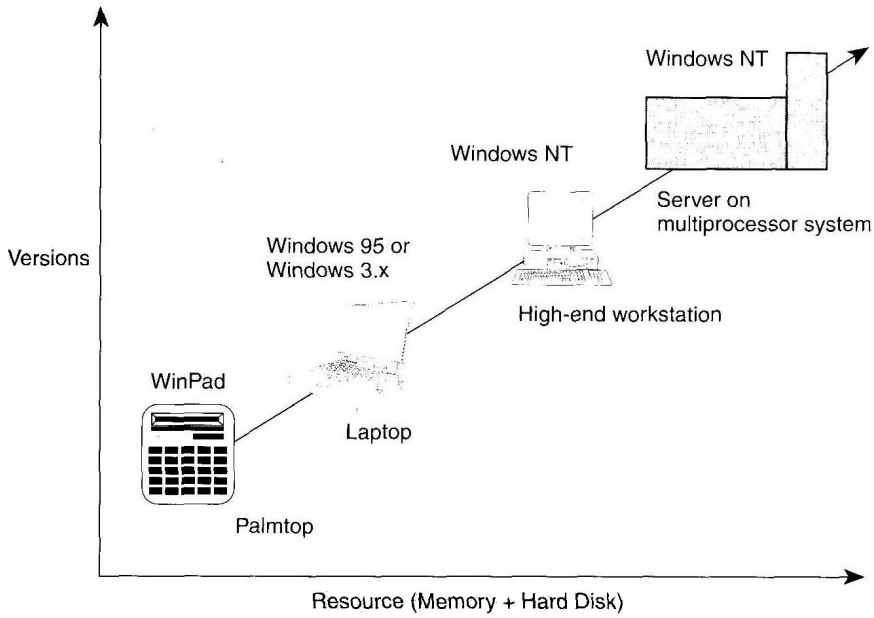


Figure 1-2. *Windows and Window NT on Different Types of Hardware*

power users use Windows NT for their computing-intensive applications, such as real-time process control, stock-market financing and forecasting, and important applications such as file servers and database servers.

1.3 IMPACT OF WINDOWS NT DESIGN ON DEVELOPERS

Windows NT can run a multitude of applications, each of which depends on different interfaces—Win16 API, Win32 API, OS/2 API, MS-DOS API, POSIX API, and Windows NT API. This flexibility is illustrated in Figure 1-3.

Win16 Application Programming Interfaces: These interfaces ensure all Windows applications written to 16-bit Windows APIs can be executed on Windows NT. (Nothing undermines the success of an operating system more than not having good applications for it, so this is a big boost to Windows NT.) This is especially important for the ISVs who currently produce Windows applications, because they can continue to meet their customer needs even if the customers move to the Windows NT platform. Chapter 2, "Understanding Windows NT Architecture," explains in detail the mechanism by which the Win16 interfaces are supported on Windows NT.

Win32 Application Programming Interfaces: Windows NT is a 32-bit operating system. To take full advantage of its power, Windows NT exports 32-bit versions of Win16 interfaces. (Of course, an application written to Win32 APIs will perform better than their Win16 counterparts.) At run time, a process called "thunking" maps Win16 API calls to Win32 calls. To make it

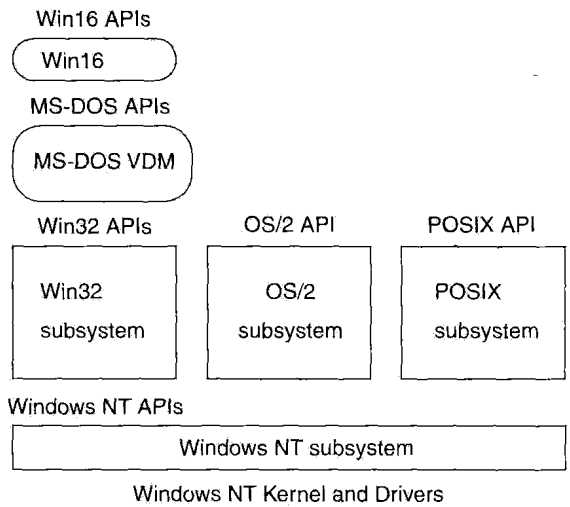


Figure 1-3. *Windows NT Programming Interfaces*

easy to gradually port Win16 applications to a Win32 environment, the arguments and return types of Win32 interfaces are designed to be almost identical to those of the Win16 interfaces, although most arguments and return types are 32-bit quantities in the Win32 environment. It is therefore possible to create a Win32 application from a Win16 application by recompiling the source, provided the original application is a well-written, "well-behaved" Win16 application. In practice, some changes might be necessary, but they will be minimal.

Win32 interfaces also provide advanced operating system features, such as semaphores, events, and threads. Hence, it is prudent to implement enhancements in the application so it uses the advanced features given by Win32 APIs. Chapter 2, "Understanding Windows NT Architecture," has more details.

MS-DOS Application Programming Interfaces: Windows NT supports most MS-DOS calls (for example, INT 21h), so most of the popular MS-DOS applications can be run on Windows NT. Some MS-DOS interfaces are not allowed in Windows NT.[5] These include the interfaces that allow applications to directly interact with the hardware, such as the video or the hard disk.

OS/2 Application Programming Interfaces: Windows NT supports most OS/2 APIs, including console-based **I/O (input/output)** APIs. Currently, it does not support **Presentation Manager™ (PM)** interfaces, but Microsoft expects to make PM subsystems available in the near future.[6]

5. See [MS 93a] for details.
6. See [MS 93a] for details on the OS/2 APIs supported by Windows NT.