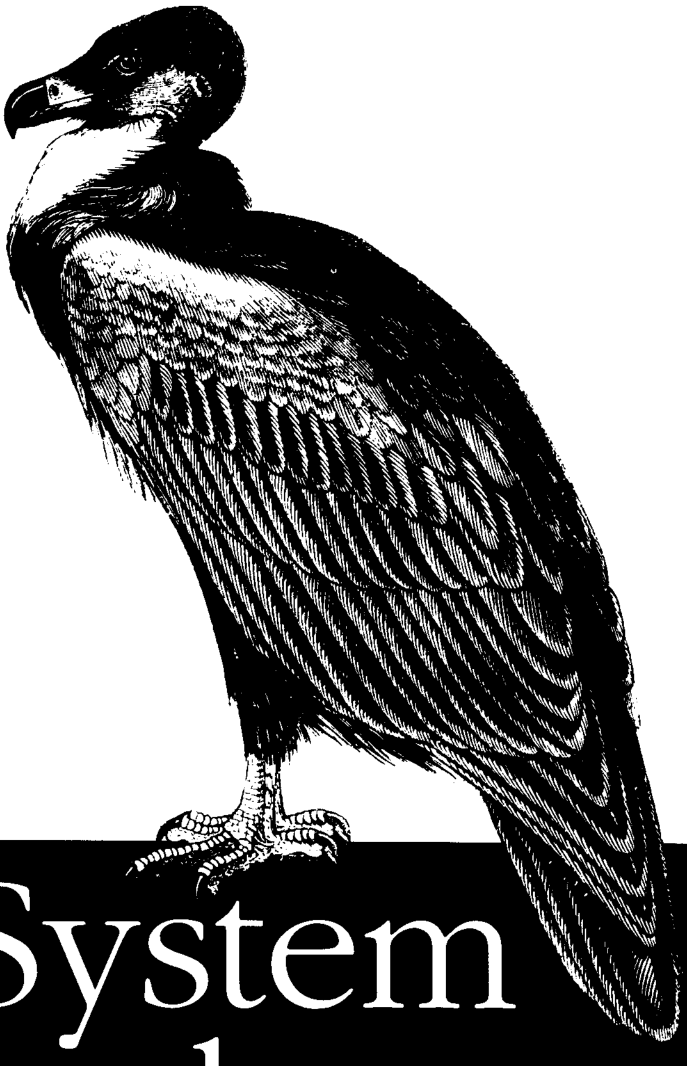
Building NT File System Drivers



Windows NT

File System Internals

A Developer's Guide

O'REILLY™

Rajeev Nagar

Windows NT File System Internals

This book is dedicated to:

My parents, Maya and Yogesh

My wife and bestfriend, Priya

Our beautiful daughters, Sana and Ria

For it is their faith, support, and encouragement

that inspires me to keep striving

Table of Contents

Preface

Over the past three years, Windows NT has come to be regarded as a serious, stable, viable, and highly competitive alternative to most other commercially available operating systems. It is also one of the very few *new* commercially released operating systems that has been developed more or less from scratch in the last 15 years, and can claim to have achieved a significant amount of success. However, Microsoft has not yet documented, in any substantial manner, the guts of this increasingly important platform. This has resulted in a dearth of reliable information available on the internals of the Windows NT operating system.

This book focuses on explaining the internals of the Windows NT I/O subsystem, the Windows NT Cache Manager, and the Windows NT Virtual Memory Manager. In particular, it focuses on file system driver and filter driver implementation for the Windows NT platform, which often requires detailed information about the above-mentioned components.

Intended Audience

This book is intended for those who have a need *today* for understanding a significant portion of the Windows NT operating system, and also for those among us who simply are curious about what makes Windows NT tick.

Typically, the book should be interesting and useful to you if you design or implement kernel-mode software, such as file system or device drivers. It should also be interesting to those of you who are studying or teaching operating system design and wish to understand the Windows NT operating system a little bit better. Finally, if you are a system administrator who *really* wants to know what it is that you have just spent the vast majority of your annual budget on (operating system licenses, additional third-party driver licenses for virus-checking software, and so on), this book should help satisfy your curiosity.

The approach taken in writing this book is that the information provided should give you *more* than what you can get from any other documentation that is currently available. Therefore, I expend a lot of effort discussing the whys and hows that underlie the design and implementation of the Windows NT I/O subsystem, Virtual Memory Manager, and Cache Manager. For those of you who need to implement a file system or filter driver module right this minute, there is a substantial amount of code included that should get you well along on your way.

Above all, this book is intended as a guide and reference to assist you in understanding a major portion of the Windows NT operating system better than you do today. I hope it will help to make you more informed about the operating system itself, which in turn should help you exploit the operating-system-provided functionality in an optimal manner.

Windows NT File System Internals was written with certain assumptions in mind: I assume that you understand the fundamentals of operating systems and therefore, do not need me to explain what an operating system is; at the same time, I do not assume that you understand file system technology (especially on the Windows NT platform) in any great detail, although such understanding will undoubtedly help you if and when you decide to design and implement a file system yourself. I further assume that you know how to develop programs using a high-level language such as C. Finally, I assume that you have some interest in the subject matter of this book; otherwise, I find it hard to imagine why anyone would want to subject themselves to more than 700 pages of excruciatingly detailed information about the I/O subsystem and associated components.

Book Contents and Organization

In order to design and develop complex software such as file system drivers or other kernel-mode drivers, it becomes necessary to first understand the operating system environment thoroughly. At the same time, I always find it useful to have sample code to play with that can assist me when I start designing and developing my own software modules. Therefore, I have organized this book along the following lines.

Part 1: Overview

This part of the book provides you with the required background material that is essential to successfully designing and developing Windows NT kernel-mode drivers. This portion of the book should be of particular interest to those of you who intend to actually develop kernel-mode software for the Windows NT platform.

Chapter 1, Windows NT System Components

This chapter provides an introduction to the various components that together constitute the kernel-mode portion of the Windows NT operating system. The overall architecture of the operating system is discussed, followed by a brief discussion on the Windows NT Kernel and the Windows NT Executive components.

Chapter 2, File System Driver Development

This chapter provides an introduction to file system and filter drivers. Some common driver development issues that arise when designing for the Windows NT platform are also discussed here, including a discussion on allocating and freeing kernel memory, working efficiently with linked lists of structures, and using Unicode strings in your driver. Finally, discussions on the Windows NT object name space and the MUP and MPR components, which are of interest to developers who wish to design network redirectors, are presented in this chapter.

Chapter 3, Structured Driver Development

Designing well-behaved kernel-mode software is the focus of this chapter. Exception dispatching support provided by the operating system is discussed here; the section on structured exception handling discusses how you can develop robust kernel-mode software. There is also a detailed discussion of the various synchronization primitives that are available to kernel-mode developers, and which are essential to writing correct system software. The synchronization primitives discussed here include spin locks, dispatcher objects, and read-write locks.

Part 2: The Managers

Part 2 of this book describes the Windows NT I/O Manager, the Windows NT Virtual Memory Manager, and the Windows NT Cache Manager in considerable detail from the perspective of a developer who wishes to design and implement file system drivers. Regardless of whether or not you eventually choose to design and implement kernel-mode software for the Windows NT platform, these chapters should be useful to you and will provide you with a detailed understanding of some important and complex Windows NT operating system software modules.

Chapter 4, The NT I/O Manager

This chapter takes a detailed look at the Windows NT I/O Manager. The components of the I/O subsystem, as well as the design principles that guided the development of the I/O Manager and I/O subsystem components, are discussed here; so is the concept of *thread-context,* which is extremely important for kernel-mode driver developers. This chapter also provides a description of some of the more important system data structures and of handling synchronous and asynchronous I/O requests. Finally, a high-level overview of the operating system boot sequence is included.

Chapter 5, The NT Virtual Memory Manager

Topics discussed in this chapter include the functionality provided by the VMM, process address space layout, physical memory management and virtual address space manipulation support provided by the Virtual Memory Manager, and memory-mapped file support. This chapter provides an overview on how page fault handling is provided by the VMM, on the workings of the modified page writer, and finally, on the interactions of the Virtual Memory Manager with file system drivers.

Chapter 6, The NT Cache Manager I

This chapter provides an introduction to the Windows NT Cache Manager. The functionality provided by the Cache Manager is discussed here, followed by a discussion on how cached read and write I/O requests are jointly handled by the I/O Manager, file system drivers, and the Cache Manager. The various Cache Manager interfaces are introduced, followed by a discussion on the clients that typically request services from the Windows NT Cache Manager. Some important data structures required for successful interaction with the Cache Manager are also described. Finally, there is a discussion on how file size manipulation can be successfully performed for cached files.

Chapter 7, The NT Cache Manager II

This chapter provides an overview of how the Windows NT Cache Manager uses internal data structures to provide caching services to the rest of the system. File system drivers must be cognizant of certain requirements that they must fulfill to interact successfully with the Cache Manager; these requirements are discussed here. This chapter also has details of each of the various interfaces (function calls) that are available to Cache Manager clients.

Chapters, The NT Cache Manager III

Topics discussed in this chapter include flushing the system cache, terminating caching for a file, descriptions of certain miscellaneous Cache-Manager-provided function calls, and the interactions of the Cache Manager with the I/O Manager, and the Virtual Memory Manager. Finally, read-ahead and delayed-write functionality, provided by the Windows NT Cache Manager, is discussed.

Part 3: The Drivers

Part 3 describes how to use the information provided in Parts 1 and 2 of this book. This portion of the book focuses exclusively on actual design and development of two types of kernel-mode drivers. It could also be used as a reference in understanding how the various Windows NT file systems process user requests for file I/O and as an aid to understanding what is actually going on in the system when you debug any lower-level kernel-mode driver that you may have developed.

Chapter 9, Writing a File System Driver I
This chapter provides an introduction to file system design and also describes how to configure (via Registry entries) your file system driver implementation on a Windows NT system. A comprehensive description of the important data structures that you should implement in order to develop a Windows NT file system driver is also provided. Details on how you can implement the create/open, read, and write dispatch routines in your file system driver are included.

Chapter 10, Writing A File System Driver II
This chapter contains discussions of some important concepts that you should understand when trying to design a Windows NT file system driver; these include the concept of the top-level component for an IRP and how to implement support for asynchronous I/O requests in your file system driver. A description of how to implement support for processing the directory control, cleanup, and close requests is also provided.

Chapter 11, Writing a File System Driver III
Topics discussed in this chapter include the fast I/O method for data access, implementing callback routines in your FSD for use by the Windows NT Cache Manager and Virtual Memory Manager, dispatch routines including flushing file buffers, getting and setting volume information, implementing byte-range lock support, supporting opportunistic locking, and implementing support for file system control and device I/O control requests (including a detailed discussion on handling mounting and verification requests for logical volumes). Finally, there is a detailed discussion of how to implement a mini-file system recognizer driver for your file system driver product.

Chapter 12, Filter Drivers
A description of the functionality that can be provided by a filter driver is followed by some examples of customer requirements where filter driver development can be useful. Topics discussed here include getting a pointer to the appropriate target device, attaching to the target device object, the consequences of executing an attach operation, and the various I/O-Manager-provided support functions available for use by a filter driver.

Appendixes

Appendix A, Windows NT System Services
This appendix contains a detailed listing of the major Windows NT I/O Manager-provided *native* system calls.

Appendix B, MPR Support
This appendix describes functions that network redirectors should implement to provide MPR support.

Appendix C, Building Kernel-Mode Drivers
This appendix provides an overview of the build process used to create kernel-mode drivers.

Appendix D, Debugging Support
An introduction to the Microsoft *WinDbg* source-level debugger is provided.

Appendix E, Recommended Readings and References
A list of recommended readings is provided for your benefit if you wish to delve further into or get more detailed information on some of the topics discussed in this book.

Appendix F, Additional Sources for Help
This appendix lists some online sources and other resources that you can explore for more information on kernel-mode development for Windows NT.

I would suggest that the chapters be read in the sequence in which they are organized. However, advanced readers who understand the basic kernel-mode environment on the Windows NT platform may wish to skip directly to Part 2 of this book. Throughout this book, an effort has been made to avoid forward references to undefined terms; however, such references are flagged whenever they cannot be avoided.

Accompanying Diskette

A diskette accompanies this book and is often referred to in various chapters of the book. This diskette contains source code for the following:*

A file system driver template
Note carefully that this is simply a skeleton driver that does not provide for most of the functionality typically implemented by file system drivers. The code has been compiled for the Intel x86 platform. The code has not been tested, however, and should never be used as is without major enhancement and testing efforts on your part.

* Many of the file system dispatch routines arc also documented and discussed in the text.

This driver source is provided as a framework for you to use to design and implement a real file system driver for the Windows NT environment.

A filter driver implementation

The filter driver for which source has been provided intercepts all I/O requests targeted to a specified mounted logical volume. You can extend this filter driver source code to implement any value-added functionality you wish to provide to your customers.

If you intend to develop kernel-mode software for the Windows NT platform, I strongly recommend that you obtain at least a *Professional Level Subscription* to the Microsoft Developer's Network (MSDN). This subscription will provide you with access to the Windows NT Device Driver's Kit (DDK), associated documentation, and a reasonable number of additional benefits. Contact the Microsoft Developer's Network at *http://www.microsoft.com/msdn* for additional details.

Note that the source code provided on the accompanying disk has only been compiled using the Microsoft Visual C++ compiler (Version 4.2). This compiler can be purchased directly from Microsoft. They can be reached on the World Wide Web at *http://www.microsoft.com/visualc.*

Finally, you should note that successful compilation of the file system driver source requires a header file (`ntifs.h`) that is currently only available from Microsoft by purchasing a Windows NT IPS kit. This kit was released in April 1997 and is sold as a separate product by Microsoft from the MSDN subscription. You can obtain more information about this product at *http://www.microsoft.com/hwdev/ntifskit.* Although many of the structure, constant, and type definitions contained in the header file have been provided in this book, they are subject to frequent change, and I would encourage you to carefully evaluate your requirements and try to purchase this product if at all possible.

Conventions Used in This Book

This book uses the following font conventions:

Italic

is used for World Wide Web URL addresses, to display email addresses, to display Usenet newsgroup addresses, and to highlight special terms the first time they are defined.

`Constant Width`

is used to display command names, filenames, field names, constant definitions, type (structure) definitions, and in code examples.

Acknowledgments

I consider myself extremely fortunate to have known, studied under, and worked with some of the most exceptional minds in the field of computer science. I would like to especially thank the following individuals: for introducing me to synchronization primitives and serving as an advisor to a much-harried and perennially late-to-complete-thesis graduate student, Dr. Sheau-Dong Lang at the University of Central Florida. Also, Dr. Ronald Dutton of the University of Central Florida for teaching me the fundamentals of algorithm design and analysis, and supporting me through one of the most difficult periods in my academic life. I would like to acknowledge the trust, support, and friendship of Robert Smith, whom I consider a mentor and friend and who entrusted me, a rookie engineer, to write his first commercial file system driver. I would also like to thank my colleagues at each of the companies I have worked at, namely, Micro Design International, Inc., Transarc Inc., and Hewlett-Packard Inc., for their support and advice. My grateful thanks to our technical reviewers: Mike Kazar, Derrel Blain, and David J. Van Maren who took time from their busy schedules to review this book.

Many thanks to the people at O'Reilly and Associates who have contributed to this effort: Mary Anne Weeks Mayo was the production project manager and quality was assured by Ellie Fountain Maden, John Files, Nicole Gipson Arigo, and Sheryl Avruch. Seth Maislin wrote the index. Madeline Newell and Colleen Miceli lent critical freelance support. Mike Sierra contributed his FrameMaker tool-tweaking prowess. Chris Reilley and Robert Romano were responsible for the crisp illustrations you see in the book. The book's interior was designed by Nancy Priest, Edie Freedman designed the front cover, and Hanna Dyer designed the back cover.

Finally, many thanks to Robert Denn for his editorial support over the past year and, most importantly, for his patience and trust that I would eventually complete this effort. It has been a pleasure working with him.