Introduction to Windows File System Development

Tony Mason

Abstract

File Systems Development on Windows can be somewhat different than similar development on Linux or UNIX systems, both in terms of the range of options as well as the development environment. Given the prevalence of Windows systems, however, developing a file system yields similar benefits for Windows as it does for other systems because it provides common services that are accessible to a broad range of applications.

Using a file systems development perspective, this tutorial will provide a basic overview of the Operating Systems Structure: Kernel, I/O Subsystem, Virtual Memory Manager, and Object Manager. Because Windows naming differs from traditional POSIX operating systems, the tutorial will discuss the name space, including support for 16-bit UNICODE characters, the maximum 32K character path/file name - including misconceptions caused by the Win32 programming paradigm.

Additionally, the tutorial will discuss the Windows kernel mode driver model, which allows dynamic loading and unloading of file system drivers. It will discuss three distinct types of file systems that are commonly constructed for Windows: physical media file systems, network file systems, and virtual (pseudo) file systems. It will also discuss the commonality and differences between the VFS interface and the Windows interface, with an eye towards cross-platform support.

Further, the tutorial will discuss how Windows supports *layered* file systems through the use of "file system filter drivers"; while often thought of as simpler to build, the tutorial will discuss the pragmatic reality that they can be more challenging to develop than traditional file system drivers.

In addition, the tutorial will describe the rich development environment available for building file systems on Windows - all of which are available without charge - including the integrated development environment (IDE), debuggers (kd/WindBG), installation, and testing. Because Windows does require cryptographically signed drivers, the tutorial will describe the basics of this process.

Finally, the tutorial will briefly describe FUSE on Windows, an alternative to constructing a kernel mode file system driver, which is highly compatible with the FUSE interface on UNIX and Linux systems.

This tutorial provides an introduction to the Windows kernel environment from a file systems development perspective. In addition to describing the basic structure of the Windows kernel, it will also explain the range of options for file systems development, including developing traditional network, media, and pseudo file systems as well as the layered file systems model ("file system filter drivers"). Because the Windows kernel interface is substantially broader than the typical "Win32" interface with which most people are familiar, the tutorial will highlight key differences. The tutorial will explore the rich development environment: the tools needed to build file systems on Windows, including debugger, compiler, and integrated development environment (IDE), as well as approaches to testing, validating, and signing file system drivers. Finally, it will briefly explore issues in porting file systems between Linux/UNIX systems and Windows systems.

Topics include:

- Windows kernel architecture
- I/O Subsystem
- Virtual Memory Subsystem
- Name space management
- File Systems Drivers
- File System Filter Drivers
- Development Tools
- Building, testing, validating, and deploying File System drivers
- FUSE on Windows as an alternative

Author

Tony Mason is a PhD Student at the University of British Columbia, focusing on non-hierarchical file system name spaces and persistent memory. He has been developing file systems for the past 29 years, including on UNIX and Windows systems. Since 1996, he has taught numerous courses in Windows kernel mode development, including device drivers, kernel debugging, file systems, and file system filter driver development. Tony is co-author on *Windows NT Device Driver Development* [1] and *lex & yacc* [2].

Bibliography

[1] P. Viscarola and W. A. Mason, *Windows NT device driver development.* New Riders Publishing, 1998.

[2] J. R. Levine, J. R. Levine, T. Mason, and D. Brown, *Lex & yacc.* " O'Reilly Media, Inc.", 1992.