INSIDE WINDOWS NT™

TM

Microsoft PRESS®

HELEN CUSTER

FOREWORD BY DAVID N. CUTLER

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Library of Congress Cataloging-in-Publication Data
Custer, Helen, 1961–
Inside windows NT / Helen Custer.
p. cm.
Includes bibliographical references and index.
ISBN 1-55615-481-X
1. Operating systems (Computers) 2. Windows NT. I. Title.
QA76.76.063C89 1992
005.4'469--dc20 92-26231
CIP

Printed and bound in the United States of America.

4 5 6 7 8 9 AGAG 8 7 6 5 4 3

Distributed to the book trade in Canada by Macmillan of Canada, a division of Canada Publishing Corporation.

Distributed to the book trade outside the United States and Canada by Penguin Books Ltd.

Penguin Books Ltd., Harmondsworth, Middlesex, England
Penguin Books Australia Ltd., Ringwood, Victoria, Australia
Penguin Books N.Z. Ltd., 182–190 Wairau Road, Auckland 10, New Zealand

British Cataloging-in-Publication Data available.

Acquisitions Editor: Dean Holmes
Manuscript Editor: Nancy Siadek
Project Editors: Nancy Siadek and Deborah Long
Technical Editor: Jeff Carey

The object named Floppy0 is a device object, a special object type defined and used by the I/O system. In the object manager namespace, the device object represents a launching point into a file system's object domain, one that the object manager knows nothing about.

When the I/O system created the device object type, it registered a parse method for it. When the object manager looks up an object name, it suspends its search when it encounters an object in the path that has an associated parse method. The object manager calls the parse method, passing to it the remainder of the object name it is looking for.

For example, when a process opens a handle to the object named \Device\Floppy0\docs\resume.doc, the object manager traverses its name tree until it reaches the device object named Floppy0. It sees that a parse method is associated with this object, and it calls the method, passing to it the rest of the object name it was searching for—in this case, the string \docs\resume.doc. The parse method for device objects is an I/O routine. The routine takes the name string and passes it to the appropriate file system, which finds the file on the disk and opens it.

The symbolic link objects described in Section 3.2.1.3 are also translated by a parse method. The symbolic link object type has a parse method associated with it. The method takes one name, substitutes another name for it, and then calls the object manager to restart its search for the object. (If the new name also contains a symbolic link object name, the parse method is called again.)

The security method, which is used by the I/O system, is similar to the parse method. It is called whenever a thread tries to change the security information protecting a file. This information is different for files than for other objects because security information is stored in the file itself rather than in memory. The I/O system, therefore, must be called in order to find the security information and change it.

3.3 Protecting Objects

Although naming, sharing, and accounting for system resources in a uniform way are all good reasons for the NT executive to use an object model, probably the most important reason is to ensure that Windows NT is a secure operating system.

Operating system security is a battle fought on many fronts. A secure multiuser system must protect one user's files, memory, and other resources from other users. It must protect the operating system's data, files, and memory from user programs. It should monitor attempts to bypass its security

an operating system that make it secure. These features are categorized into seven levels of security, each one more stringent than the last.[7]

At the Class C2 level, the initial target for Windows NT, the following features must be present:

- A *secure logon facility* requires users to identify themselves by entering a unique logon identifier and a password before they are allowed access to the system.
- *Discretionary access control* allows an owner of a resource to determine who can access the resource and what they can do to it. The owner does this by granting access rights to a user or a group of users.
- *Auditing* provides the ability to detect and record important security-related events or any attempt to create, access, or delete system resources. It uses logon identifiers to record the identity of the user who performed the action.
- *Memory protection* prevents anyone from reading information written by someone else after a data structure has been released back to the operating system. Memory is reinitialized before it is reused.

Not all Windows NT installations will require all the security mechanisms that the system provides. The security system, therefore, allows a system administrator to streamline the logon sequence, for example, or to adjust whether information is collected in an audit log and, if so, how much.

Facilities that are extremely security conscious, such as military installations, require an even higher level of security than Windows NT initially provides. Therefore, Windows NT is designed to evolve toward Class B2 security, a level known as Mandatory Access Control, in which each user is assigned a security clearance level and is prevented from giving lower-level users access to protected resources. For example, in secure U.S. government facilities, one user might have a "Secret" security clearance and another a "Top Secret" security clearance. Mandatory access control ensures that the user with the "Top Secret" clearance can never allow the former user access to any "Top Secret" information, even by using discretionary access control. Similarly, B2 security requires the recognition of "compartments," the separating of groups of users from one another. This type of protection is useful in industries such as financial security exchanges, in which inappropriate access to stock offerings or mergers might create conflicts of interest.

The Windows NT security system is multifaceted, but protecting objects is the essence of discretionary access control and auditing (and later, of mandatory access control). The idea behind Windows NT security is to create a gate through which every user of system resources must pass. Because all system resources that can be compromised are implemented as objects, the NT object manager becomes the gate. One need not poke around in numerous dark corners of the operating system to validate the integrity of Windows NT's security system; the critical security-related operations can be found in a central location.

The following subsections examine object protection from two perspectives: first, verifying the identity of users and, second, controlling which users can access which objects.

3.3.1 Access Tokens

In order to control who can manipulate an object, the security system must first be sure of each user's identity. Therefore, the first line of protection in Windows NT is the requirement that every user log onto the system.

As Chapter 2, "System Overview," described, an integral protected subsystem, the *security subsystem*, is responsible for *authenticating* users—that is, for verifying that the logon information a user supplies matches the information stored in a security database. After the security subsystem determines that a logon is authentic, it constructs an object that it permanently attaches to the user's process. This object is called an *access token*, and it serves as the process's official identity card whenever it tries to use a system resource. A sample access token is depicted in Figure 3-8.

The first attribute shown in this example is the user's personal *security ID*, an identifier that usually corresponds to the user's logon name. In large installations, a security ID might also incorporate the name of the user's divi-

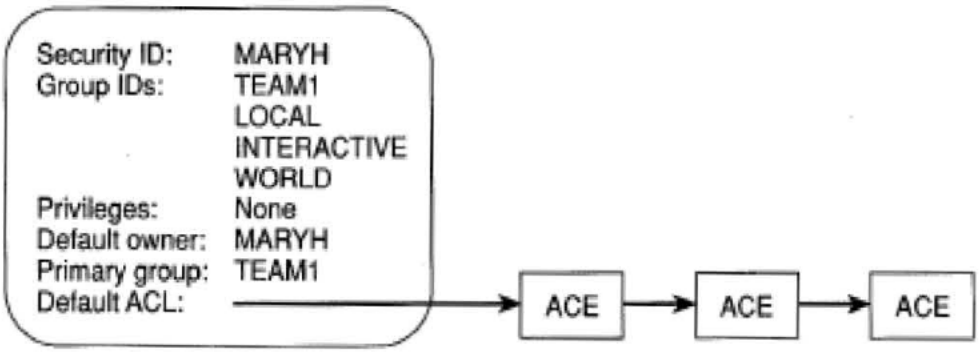

Figure 3-8. Sample Access Token

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.