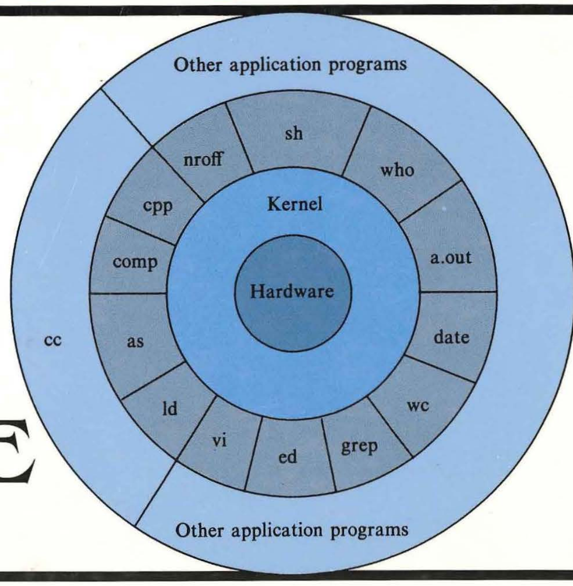
THE DESIGN OF THE UNIX® OPERATING SYSTEM



MAURICE J. BACH

PRENTICE-HALL SOFTWARE SERIES

BACH THE DESIGN OF THE UNIX®

THE DESIGN OF THE UNIX® OPERATING SYSTEM

Maurice J. Bach

PRENTICE-HALL, INC., Englewood Cliffs, New Jersey 07632

Published by Prentice-Hall, Inc.
A division of Simon & Schuster
Englewood Cliffs, New Jersey 07632

Prentice-Hall Software Series
Brian W. Kernighan, Advisor

Cover Design Consultant: Sarah Bach

UNIX® is a registered trademark of AT&T.
DEC, PDP, and VAX are trademarks of Digital Equipment Corp.
Series 32000 is a trademark of National Semiconductor Corp.
®Ada is a registered trademark of the U.S. Government (Ada Joint Program Office).
UNIVAC is a trademark of Sperry Corp.
This document was set on an AUTOLOGIC, Inc. APS-5 phototypesetter driven by the TROFF formatter operating under the UNIX system on an AT&T 3B20 computer.

The Publisher offers discounts on this book when ordered in bulk quantities. For more information write:

Special Sales/College Marketing
Prentice-Hall, Inc.
College Technical and Reference Division
Englewood Cliffs, New Jersey 07632

Printed in the United States of America

20 19 18 17 16 15 14 13 12 11

ISBN 0-13-201799-7 025

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Prentice-Hall of Southeast Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

There are several forms of interprocess communication, ranging from asynchronous signaling of events to synchronous transmission of messages between processes.

Finally, the hardware control is responsible for handling interrupts and for communicating with the machine. Devices such as disks or terminals may interrupt the CPU while a process is executing. If so, the kernel may resume execution of the interrupted process after servicing the interrupt: Interrupts are *not* serviced by special processes but by special functions in the kernel, called in the context of the currently running process.

2.2 INTRODUCTION TO SYSTEM CONCEPTS

This section gives an overview of some major kernel data structures and describes the function of modules shown in Figure 2.1 in more detail.

2.2.1 An Overview of the File Subsystem

The internal representation of a file is given by an *inode*, which contains a description of the disk layout of the file data and other information such as the file owner, access permissions, and access times. The term inode is a contraction of the term *index node* and is commonly used in literature on the UNIX system. Every file has one inode, but it may have several names, all of which map into the inode. Each name is called a *link*. When a process refers to a file by name, the kernel parses the file name one component at a time, checks that the process has permission to search the directories in the path, and eventually retrieves the inode for the file. For example, if a process calls

```
open("/fs2/mjb/rje/sourcefile", 1);
```

the kernel retrieves the inode for "/fs2/mjb/rje/sourcefile". When a process creates a new file, the kernel assigns it an unused inode. Inodes are stored in the file system, as will be seen shortly, but the kernel reads them into an in-core[1] inode table when manipulating files.

The kernel contains two other data structures, the *file table* and the *user file descriptor table*. The file table is a global kernel structure, but the user file descriptor table is allocated per process. When a process *open*s or *creat*s a file, the kernel allocates an entry from each table, corresponding to the file's inode. Entries in the three structures — user file descriptor table, file table, and inode table — maintain the state of the file and the user's access to it. The file table keeps track of the byte offset in the file where the user's next *read* or *write* will start, and the

1. The term *core* refers to primary memory of a machine, not to hardware technology.

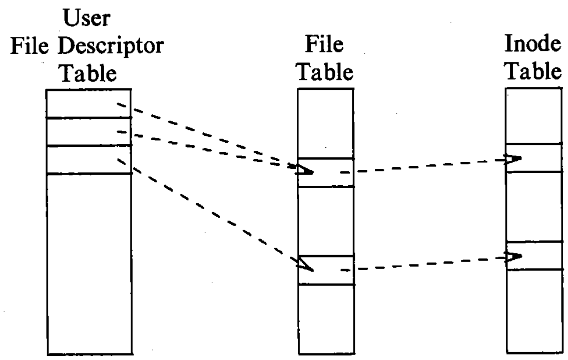


Figure 2.2. File Descriptors, File Table, and Inode Table

access rights allowed to the *open*ing process. The user file descriptor table identifies all open files for a process. Figure 2.2 shows the tables and their relationship to each other. The kernel returns a *file descriptor* for the *open* and *creat* system calls, which is an index into the user file descriptor table. When executing *read* and *write* system calls, the kernel uses the file descriptor to access the user file descriptor table, follows pointers to the file table and inode table entries, and, from the inode, finds the data in the file. Chapters 4 and 5 describe these data structures in great detail. For now, suffice it to say that use of three tables allows various degrees of sharing access to a file.

The UNIX system keeps regular files and directories on block devices such as tapes or disks. Because of the difference in access time between the two, few, if any, UNIX system installations use tapes for their file systems. In coming years, diskless work stations will be common, where files are located on a remote system and accessed via a network (see Chapter 13). For simplicity, however, the ensuing text assumes the use of disks. An installation may have several physical disk units, each containing one or more *file systems*. Partitioning a disk into several file systems makes it easier for administrators to manage the data stored there. The kernel deals on a logical level with file systems rather than with disks, treating each one as a *logical device* identified by a logical *device number*. The conversion between logical device (file system) addresses and physical device (disk) addresses is done by the disk driver. This book will use the term device to mean a logical device unless explicitly stated otherwise.

A file system consists of a sequence of logical blocks, each containing 512, 1024, 2048, or any convenient multiple of 512 bytes, depending on the system implementation. The size of a logical block is homogeneous within a file system but may vary between different file systems in a system configuration. Using large logical blocks increases the effective data transfer rate between disk and memory,

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.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase®
Smarter legal research.