



附光盘

大学计算机教育丛书（影印版）

Second Edition

OPERATING SYSTEMS

Design and Implementation

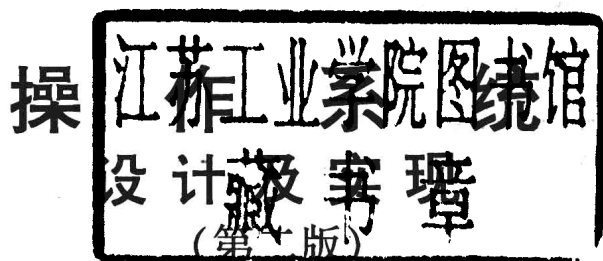
操作系统 设计及实现 (第二版)

Andrew S. Tanenbaum
Albert S. Woodhull

清华大学出版社 · PRENTICE HALL

OPERATING SYSTEMS:

Design and Implementation
Second Edition



Andrew S. Tanenbaum
Albert S. Woodhull

清华大学出版社
Prentice-Hall International Inc.

(京)新登字 158 号

Operating Systems: design and implementation 2nd Ed. /Andrew S. Tanenbaum, Albert S. Woodhull

© 1997 by Prentice Hall, Inc.

Original edition published by Prentice Hall, Inc., a Simon & Schuster Company.

Prentice Hall 公司授权清华大学出版社在中国境内(不包括香港、澳门和台湾)独家出版发行本书影印本。

本书任何部分内容,未经出版者书面同意,不得用任何方式抄袭、节录或翻印。

本书封面贴有 Prentice Hall 激光防伪标签,无标签者不得销售。

北京市版权局著作权合同登记号: 01-97-1230

图书在版编目(CIP)数据

操作系统:设计及实现:第二版:英文/(美)特南鲍姆(Tanenbaum A.S), (美)伍德哈尔(Woodhull A.S)著. - 影印本. - 北京:清华大学出版社,1997.9
(大学计算机教育丛书)

ISBN 7-302-02714-5

I. 操… II. ①特… ②伍… III. 操作系统(软件) IV. TP316

中国版本图书馆 CIP 数据核字(97)第 17357 号

出版者:清华大学出版社(北京清华大学校内,邮编 100084)

因特网地址: www.tup.tsinghua.edu.cn

印刷者:清华大学印刷厂

发行者:新华书店总店北京科技发行所

开 本: 850×1168 1/32 印张: 30

版 次: 1997 年 9 月第 1 版 1998 年 7 月第 3 次印刷

书 号: ISBN 7-302-02714-5/TP·1405

印 数: 10001~15000

定 价: 69.00 元

出版前言

我们的大学生、研究生毕业后,面临的将是一个国际化的信息时代。他们将需要随时查阅大量的外文资料;会有更多的机会参加国际性学术交流活动;接待外国学者;走上国际会议的讲坛。作为科技工作者,他们不仅应有与国外同行进行口头和书面交流的能力,更为重要的是,他们必须具备极强的查阅外文资料获取信息的能力。有鉴于此,在国家教委所颁布的“大学英语教学大纲”中有一条规定:专业阅读应作为必修课程开设。同时,在大纲中还规定了这门课程的学时和教学要求。有些高校除开设“专业阅读”课之外,还在某些专业课拟进行英语授课。但教、学双方都苦于没有一定数量的合适的英文原版教材作为教学参考书。为满足这方面的需要,我们挑选了7本计算机科学方面最新版本的教材,进行影印出版。Prentice Hall 公司和清华大学出版社这次合作将国际先进水平的教材引入我国高等学校,为师生们提供了教学用书,相信会对高校教材改革产生积极的影响。

清华大学出版社
Prentice Hall 公司

1996. 11

**OPERATING SYSTEMS:
DESIGN AND IMPLEMENTATION**

Second Edition

PREFACE

Most books on operating systems are strong on theory and weak on practice. This one aims to provide a better balance between the two. It covers all the fundamental principles in great detail, including processes, interprocess communication, semaphores, monitors, message passing, scheduling algorithms, input/output, deadlocks, device drivers, memory management, paging algorithms, file system design, security, and protection mechanisms. But it also discusses one particular system—MINIX, a UNIX-compatible operating system—in detail, and even provides a complete source code listing for study. This arrangement allows the reader not only to learn the principles, but also to see how they are applied in a real operating system.

When the first edition of this book appeared in 1987, it caused something of a small revolution in the way operating systems courses were taught. Until then, most courses just covered theory. With the appearance of MINIX, many schools began to have laboratory courses in which students examined a real operating system to see how it worked inside. We consider this trend highly desirable and hope this second edition strengthens it.

It its first 10 years, MINIX has undergone many changes. The original code was designed for a 256K 8088-based IBM PC with two diskette drives and no hard disk. It was also based on Version 7 of UNIX. As time went on, MINIX evolved in many ways. For example, the current version will now run on anything from the original PC (in 16-bit real mode) to large Pentiums with massive hard disks (in 32-bit protected mode). It also changed from being based on Version 7,

to being based on the international POSIX standard (IEEE 1003.1 and ISO 9945-1). Finally, many features were added, perhaps too many in our view, but too few in the view of some other people, which led to the creation of LINUX. In addition, MINIX was ported to many other platforms, including the Macintosh, Amiga, Atari, and SPARC. This book covers only MINIX 2.0, which so far runs only on computers with an 80x86 CPU, on systems which can emulate such a CPU, or on the SPARC.

This second edition of the book has many changes throughout. Nearly all of the material on principles has been revised, and considerable new material has been added. However, the main change is the discussion of the new, POSIX-based MINIX, and the inclusion of the new code in this book. Also new is the inclusion of a CD-ROM in each book containing the full MINIX source code plus instructions for installing MINIX on a PC (see the file *README.TXT* in the main CD-ROM directory).

Setting up MINIX on an 80x86 PC, whether for individual use or for a laboratory is straightforward. A disk partition of at least 30 MB must be made for it, then it can be installed by just following the instructions in the *README.TXT* file on the CD-ROM. To print the *README.TXT* file on a PC, first start MS-DOS, if it is not already running (from WINDOWS, click on the MS-DOS) icon. Then type

```
copy readme.txt prn
```

to make the printout. The file can also be examined in *edit*, *wordpad*, *notepad*, or any other text editor that can handle flat ASCII text.

For schools (or individuals) that do not have PCs available, two other options are now available. Two simulators are included on the CD-ROM. One, written by Paul Ashton, runs on SPARCs. It runs MINIX as a user program on top of Solaris. As a consequence, MINIX is compiled into a SPARC binary and runs at full speed. In this mode, MINIX is no longer an operating system, but a user program, so some changes to the low-level code were necessary.

The other simulator was written by Kevin P. Lawton of Bochs Software Company. This simulator interprets the Intel 80386 instruction set and enough I/O gear that MINIX can run on the simulator. Of course running on top of an interpreter costs some performance, but it makes debugging much easier for students. This simulator has the advantage that it will run on any computer that supports the M.I.T. X Window System. For more information about both simulators, please see the CD-ROM.

The development of MINIX is an ongoing proposition. The contents of this book and its CD-ROM are merely a snapshot of the system as of the time of publication. For the current state of affairs, please see the MINIX home page on the World Wide Wide, <http://www.cs.vu.nl/~ast/minix.html>. In addition, MINIX has its own USENET newsgroup: *comp.os.minix*, to which readers can subscribe to find out what is going on in the MINIX world. For those with e-mail, but without newsgroup access, there is also a mailing list. Write to listserv@listserv.nodak.edu

with "subscribe minix-l <your full name>" as the first and only line in the body of the message. You will receive more information by return e-mail.

For classroom use, a problem solutions manual is available, to instructors only, from Prentice Hall. PostScript files containing all the figures in the book, suitable for making overhead sheets, can be found by following the link marked "Software and supplementary material" from <http://www.cs.vu.nl/~ast/>.

We have been extremely fortunate in having the help of many people during the course of this project. First and foremost, we would like to thank Kees Bot for doing the lion's share of the work in making MINIX conform to the standard and for managing the distribution. Without his enormous help, we would never have made it. He wrote large chunks of code himself (e.g. the POSIX terminal I/O), cleaned up other sections, and repaired numerous bugs that had crept in over the years. Thank you for a job well done.

Bruce Evans, Philip Homburg, Will Rose, and Michael Temari have all contributed to the development of MINIX over the years. Hundreds of other people have contributed to MINIX via the newsgroup. There were so many of them and their contributions have been so varied that we cannot even begin to list them all here, so the best we can do is a generic thank you to all of them.

Several people read parts of the manuscript and made suggestions. We would like to give our special thanks to John Casey, Dale Grit, and Frans Kaashoek.

A number of students at the Vrije Universiteit tested the beta version of the CD-ROM. These were: Ahmed Batou, Goran Dokic, Peter Gijzel, Thomer Gil, Dennis Grimbergen, Roderick Groesbeek, Wouter Haring, Guido Kollerie, Mark Lassche, Raymond Ris, Frans ter Borg, Alex van Ballegooy, Ries van der Velden, Alexander Wels, and Thomas Zeeman. We would like to thank all of them for their careful work and detailed reports.

ASW would also like to thank several of his former students, particularly Peter W. Young of Hampshire College and Maria Isabel Sanchez and William Puddy Vargas of the Universidad Nacional Autonoma de Nicaragua for the part their interest in MINIX played in sustaining his efforts.

Finally, we would like to thank our families. Suzanne has been through this ten times now. Barbara has been through it nine times now. Marvin has been through it eight times now. Even Little Bram has been through it four times. It's kind of getting to be routine, but the love and support is still much appreciated. (ast)

As for Al's Barbara, this is the first time she has been through this. It would not have been possible without her support, patience, and good humor. It has been Gordon's good fortune to have been away at college through most of this. But it is a delight to have a son who understands and cares about the same things that fascinate me. (asw)

Andrew S. Tanenbaum
Albert S. Woodhull

ABOUT THE AUTHORS

Andrew S. Tanenbaum has an S.B. degree from M.I.T. and a Ph.D. from the University of California at Berkeley. He is currently a Professor of Computer Science at the Vrije Universiteit in Amsterdam, The Netherlands, where he heads the Computer Systems Group. He is also Dean of the Advanced School for Computing and Imaging, an inter-university graduate school doing research on advanced parallel, distributed, and imaging systems. Nevertheless, he is trying very hard to avoid turning into a bureaucrat.

In the past, he has done research on compilers, operating systems, networking, and local-area distributed systems. His current research focuses primarily on the design of wide-area distributed systems that scale to millions of users. These research projects have led to over 70 refereed papers in journals and conference proceedings, and five books.

Prof. Tanenbaum has also produced a considerable volume of software. He was the principal architect of the Amsterdam Compiler Kit, a widely-used toolkit for writing portable compilers, as well as of MINIX. Together with his Ph.D. students and programmers, he helped design the Amoeba distributed operating system, a high-performance micro-kernel-based distributed operating system. MINIX and Amoeba are now available for free for education and research via the Internet.

His Ph.D. students have gone on to greater glory after getting their degrees. He is very proud of them. In this respect he resembles a mother hen.

Prof. Tanenbaum is a Fellow of the ACM, a Senior Member of the IEEE, a member of the Royal Netherlands Academy of Arts and Sciences, winner of the 1994 ACM Karl V. Karlstrom Outstanding Educator Award, and winner of the 1997 ACM/SIGCSE Award for Outstanding Contributions to Computer Science Education. He is also listed in *Who's Who in the World*. His home page on the World Wide Web can be found at URL <http://www.cs.vu.nl/~ast/>.

Albert S. Woodhull has an S.B. degree from M.I.T. and a Ph.D. from the University of Washington. He entered M.I.T. intending to become an electrical engineer, but he emerged as a biologist. He has been associated with the School of Natural Science of Hampshire College in Amherst, Massachusetts, since 1973. As a biologist using electronic instrumentation, he started working with microcomputers when they became readily available. His instrumentation courses for science students evolved into courses in computer interfacing and real-time programming.

Dr. Woodhull has always had strong interests in teaching and in the role of science and technology in development. Before entering graduate school he taught high school science for two years in Nigeria. More recently he spent several sabbaticals teaching computer science at Nicaragua's Universidad Nacional de Ingenieria and the Universidad Nacional Autonoma de Nicaragua.

He is interested in computers as electronic systems, and in interactions of computers with other electronic systems. He particularly enjoys teaching in the areas of computer architecture, assembly language programming, operating systems, and computer communications. He has also worked as a consultant in the development of electronic instrumentation and related software.

He has many nonacademic interests as well, including various outdoor sports, amateur radio, and reading. He enjoys travelling and trying to make himself understood in languages other than his native English. His World Wide Web home page is located on a system running MINIX, at URL <http://minix1.hampshire.edu/asw/>.

CONTENTS

PREFACE

xv

1 INTRODUCTION

1

- 1.1 WHAT IS AN OPERATING SYSTEM? 3
 - 1.1.1 The Operating System as an Extended Machine 3
 - 1.1.2 The Operating System as a Resource Manager 4
- 1.2 HISTORY OF OPERATING SYSTEMS 5
 - 1.2.1 The First Generation (1945–55) Vacuum Tubes and Plugboards 6
 - 1.2.2 The Second Generation (1955–65) Transistors and Batch Systems 6
 - 1.2.3 The Third Generation (1965–1980): ICs and Multiprogramming 8
 - 1.2.4 The Fourth Generation (1980–Present): Personal Computers 12
 - 1.2.5 History of MINIX 13
- 1.3 OPERATING SYSTEM CONCEPTS 15
 - 1.3.1 Processes 15
 - 1.3.2 Files 17
 - 1.3.3 The Shell 20
- 1.4 SYSTEM CALLS 21
 - 1.4.1 System Calls for Process Management 22
 - 1.4.2 System Calls for Signaling 26
 - 1.4.3 System Calls for File Management 28
 - 1.4.4 System Calls for Directory Management 33
 - 1.4.5 System Calls for Protection 35
 - 1.4.6 System Calls for Time Management 36

1.5	OPERATING SYSTEM STRUCTURE	37
1.5.1	Monolithic Systems	37
1.5.2	Layered Systems	39
1.5.3	Virtual Machines	40
1.5.4	Client-Server Model	42
1.6	OUTLINE OF THE REST OF THIS BOOK	44
1.7	SUMMARY	44

2 PROCESSES

47

2.1	INTRODUCTION TO PROCESSES	47
2.1.1	The Process Model	48
2.1.2	Implementation of Processes	52
2.1.3	Threads	53
2.2	INTERPROCESS COMMUNICATION	57
2.2.1	Race Conditions	57
2.2.2	Critical Sections	58
2.2.3	Mutual Exclusion with Busy Waiting	59
2.2.4	Sleep and Wakeup	63
2.2.5	Semaphores	66
2.2.6	Monitors	68
2.2.7	Message Passing	72
2.3	CLASSICAL IPC PROBLEMS	75
2.3.1	The Dining Philosophers Problem	75
2.3.2	The Readers and Writers Problem	77
2.3.3	The Sleeping Barber Problem	80
2.4	PROCESS SCHEDULING	82
2.4.1	Round Robin Scheduling	84
2.4.2	Priority Scheduling	85
2.4.3	Multiple Queues	86
2.4.4	Shortest Job First	87
2.4.5	Guaranteed Scheduling	89
2.4.6	Lottery Scheduling	89
2.4.7	Real-Time Scheduling	90
2.4.8	Two-level Scheduling	92
2.4.9	Policy versus Mechanism	93

- 2.5 OVERVIEW OF PROCESSES IN MINIX 93
 - 2.5.1 The Internal Structure of MINIX 93
 - 2.5.2 Process Management in MINIX 95
 - 2.5.3 Interprocess Communication in MINIX 97
 - 2.5.4 Process Scheduling in MINIX 98
- 2.6 IMPLEMENTATION OF PROCESSES IN MINIX 98
 - 2.6.1 Organization of the MINIX Source Code 99
 - 2.6.2 The Common Header Files 102
 - 2.6.3 The MINIX Header Files 107
 - 2.6.4 Process Data Structures and Header Files 112
 - 2.6.5 Bootstrapping MINIX 120
 - 2.6.6 System Initialization 122
 - 2.6.7 Interrupt Handling in MINIX 128
 - 2.6.8 Interprocess Communication in MINIX 137
 - 2.6.9 Scheduling in MINIX 140
 - 2.6.10 Hardware-Dependent Kernel Support 142
 - 2.6.11 Utilities and the Kernel Library 145
- 2.7 SUMMARY 147

3 INPUT/OUTPUT

153

- 3.1 PRINCIPLES OF I/O HARDWARE 154
 - 3.1.1 I/O Devices 154
 - 3.1.2 Device Controllers 155
 - 3.1.3 Direct Memory Access (DMA) 157
- 3.2 PRINCIPLES OF I/O SOFTWARE 159
 - 3.2.1 Goals of the I/O Software 159
 - 3.2.2 Interrupt Handlers 161
 - 3.2.3 Device Drivers 161
 - 3.2.4 Device-Independent I/O Software 162
 - 3.2.5 User-Space I/O Software 164
- 3.3 DEADLOCKS 166
 - 3.3.1 Resources 167
 - 3.3.2 Principles of Deadlock 168
 - 3.3.3 The Ostrich Algorithm 170
 - 3.3.4 Detection and Recovery 172
 - 3.3.5 Deadlock Prevention 173
 - 3.3.6 Deadlock Avoidance 175

- 3.4 OVERVIEW OF I/O IN MINIX 179
 - 3.4.1 Interrupt Handlers in MINIX 180
 - 3.4.2 Device Drivers in MINIX 181
 - 3.4.3 Device-Independent I/O Software in MINIX 185
 - 3.4.4 User-level I/O Software in MINIX 185
 - 3.4.5 Deadlock Handling in MINIX 186
- 3.5 BLOCK DEVICES IN MINIX 187
 - 3.5.1 Overview of Block Device Drivers in MINIX 187
 - 3.5.2 Common Block Device Driver Software 190
 - 3.5.3 *The Driver Library* 193
- 3.6 RAM DISKS 195
 - 3.6.1 RAM Disk Hardware and Software 196
 - 3.6.2 Overview of the RAM Disk Driver in MINIX 197
 - 3.6.3 Implementation of the RAM Disk Driver in MINIX 198
- 3.7 DISKS 200
 - 3.7.1 Disk Hardware 200
 - 3.7.2 Disk Software 202
 - 3.7.3 Overview of the Hard Disk Driver in MINIX 208
 - 3.7.4 Implementation of the Hard Disk Driver in MINIX 211
 - 3.7.5 Floppy Disk Handling 220
- 3.8 CLOCKS 222
 - 3.8.1 Clock Hardware 223
 - 3.8.2 Clock Software 224
 - 3.8.3 Overview of the Clock Driver in MINIX 227
 - 3.8.4 Implementation of the Clock Driver in MINIX 230
- 3.9 TERMINALS 235
 - 3.9.1 Terminal Hardware 235
 - 3.9.2 Terminal Software 240
 - 3.9.3 Overview of the Terminal Driver in MINIX 249
 - 3.9.4 Implementation of the Device-Independent Terminal Driver 264
 - 3.9.5 Implementation of the Keyboard Driver 282
 - 3.9.6 Implementation of the Display Driver 288
- 3.10 THE SYSTEM TASK IN MINIX 296
- 3.11 SUMMARY 304

4 MEMORY MANAGEMENT**309**

- 4.1 BASIC MEMORY MANAGEMENT 310
 - 4.1.1 Monoprogramming without Swapping or Paging 310
 - 4.1.2 Multiprogramming with Fixed Partitions 311
- 4.2 SWAPPING 313
 - 4.2.1 Memory Management with Bit Maps 316
 - 4.2.2 Memory Management with Linked Lists 317
- 4.3 VIRTUAL MEMORY 319
 - 4.3.1 Paging 319
 - 4.3.2 Page Tables 322
 - 4.3.3 TLBs—Translation Lookaside Buffers 327
 - 4.3.4 Inverted Page Tables 330
- 4.4 PAGE REPLACEMENT ALGORITHMS 331
 - 4.4.1 The Optimal Page Replacement Algorithm 331
 - 4.4.2 The Not-Recently-Used Page Replacement Algorithm 332
 - 4.4.3 The First-In, First-Out (FIFO) Page Replacement Algorithm 333
 - 4.4.4 The Second Chance Page Replacement Algorithm 333
 - 4.4.5 The Clock Page Replacement Algorithm 334
 - 4.4.6 The Least Recently Used (LRU) Page Replacement Algorithm 334
 - 4.4.7 Simulating LRU in Software 336
- 4.5 DESIGN ISSUES FOR PAGING SYSTEMS 338
 - 4.5.1 The Working Set Model 338
 - 4.5.2 Local versus Global Allocation Policies 339
 - 4.5.3 Page Size 341
 - 4.5.4 Virtual Memory Interface 343
- 4.6 SEGMENTATION 343
 - 4.6.1 Implementation of Pure Segmentation 347
 - 4.6.2 Segmentation with Paging: MULTICS 348
 - 4.6.3 Segmentation with Paging: The Intel Pentium 352
- 4.7 OVERVIEW OF MEMORY MANAGEMENT IN MINIX 356
 - 4.7.1 Memory Layout 358
 - 4.7.2 Message Handling 361
 - 4.7.3 Memory Manager Data Structures and Algorithms 363
 - 4.7.4 The FORK, EXIT, and WAIT System Calls 367
 - 4.7.5 The EXEC System Call 368
 - 4.7.6 The BRK System Call 371
 - 4.7.7 Signal Handling 372
 - 4.7.8 Other System Calls 378

4.8	IMPLEMENTATION OF MEMORY MANAGEMENT IN MINIX	379
4.8.1	The Header Files and Data Structures	379
4.8.2	The Main Program	382
4.8.3	Implementation of FORK, EXIT, and WAIT	382
4.8.4	Implementation of EXEC	385
4.8.5	Implementation of BRK	386
4.8.6	Implementation of Signal Handling	387
4.8.7	Implementation of the Other System Calls	393
4.8.8	Memory Manager Utilities	394
4.9	SUMMARY	396

5 FILE SYSTEMS 401

5.1	FILES	402
5.1.1	File Naming	402
5.1.2	File Structure	404
5.1.3	File Types	405
5.1.4	File Access	407
5.1.5	File Attributes	408
5.1.6	File Operations	409
5.2	DIRECTORIES	410
5.2.1	Hierarchical Directory Systems	411
5.2.2	Path Names	412
5.2.3	Directory Operations	414
5.3	FILE SYSTEM IMPLEMENTATION	415
5.3.1	Implementing Files	415
5.3.2	Implementing Directories	419
5.3.3	Disk Space Management	422
5.3.4	File System Reliability	424
5.3.5	File System Performance	429
5.3.6	Log-Structured File Systems	432
5.4	SECURITY	434
5.4.1	The Security Environment	434
5.4.2	Famous Security Flaws	436
5.4.3	Generic Security Attacks	439
5.4.4	Design Principles for Security	441
5.4.5	User Authentication	442

- 5.5 PROTECTION MECHANISMS 446
 - 5.5.1 Protection Domains 446
 - 5.5.2 Access Control Lists 448
 - 5.5.3 Capabilities 450
 - 5.5.4 Covert Channels 451
- 5.6 OVERVIEW OF THE MINIX FILE SYSTEM 453
 - 5.6.1 Messages 454
 - 5.6.2 File System Layout 454
 - 5.6.3 Bit Maps 458
 - 5.6.4 I-nodes 460
 - 5.6.5 The Block Cache 461
 - 5.6.6 Directories and Paths 463
 - 5.6.7 File Descriptors 465
 - 5.6.8 File Locking 467
 - 5.6.9 Pipes and Special Files 467
 - 5.6.10 An Example: The READ System Call 469
- 5.7 IMPLEMENTATION OF THE MINIX FILE SYSTEM 470
 - 5.7.1 Header Files and Global Data Structures 470
 - 5.7.2 Table Management 474
 - 5.7.3 The Main Program 482
 - 5.7.4 Operations on Individual Files 485
 - 5.7.5 Directories and Paths 493
 - 5.7.6 Other System Calls 498
 - 5.7.7 The I/O Device Interface 501
 - 5.7.8 General Utilities 503
- 5.8 SUMMARY 503

6 READING LIST AND BIBLIOGRAPHY

507

- 6.1 SUGGESTIONS FOR FURTHER READING 507
 - 6.1.1 Introduction and General Works 507
 - 6.1.2 Processes 509
 - 6.1.3 Input/Output 510
 - 6.1.4 Memory Management 511
 - 6.1.5 File Systems 511
- 6.2 ALPHABETICAL BIBLIOGRAPHY 512

APPENDICES

A	MINIX SOURCE CODE LISTING	521
----------	----------------------------------	------------

B	INDEX TO FILES	905
----------	-----------------------	------------

C	INDEX TO SYMBOLS	909
----------	-------------------------	------------

	INDEX	925
--	--------------	------------