

GLOBAL
EDITION



Modern Operating Systems

FOURTH EDITION

Andrew S. Tanenbaum • Herbert Bos

ALWAYS LEARNING

PEARSON

ONLINE ACCESS

Thank you for purchasing a new copy of *Modern Operating Systems, Fourth Edition, Global Edition*. Your textbook includes twelve months of prepaid access to the book's Companion Website. This prepaid subscription provides you with full access to the following student support areas:

- Online Chapters
- Lab Experiments
- Online Exercises
- Simulation Exercises

Use a coin to scratch off the coating and reveal your student access code.
Do not use a knife or other sharp object as it may damage the code.

To access the *Modern Operating Systems, Fourth Edition, Global Edition*, Companion Website for the first time, you will need to register online using a computer with an Internet connection and a web browser. The process takes just a couple of minutes and only needs to be completed once.

1. Go to **www.pearsonglobaleditions.com/Tanenbaum**
2. Click on your book.
3. Click on **Companion Website**.
4. Click on the **Register** button.
5. On the registration page, enter your student access code* found beneath the scratch-off panel. Do not type the dashes. You can use lower- or uppercase.
6. Follow the on-screen instructions. If you need help at any time during the online registration process, simply click the **Need Help?** icon.
7. Once your personal Login Name and Password are confirmed, you can begin using the *Modern Operating Systems* Companion Website!

To log in after you have registered:

You only need to register for this Companion Website once. After that, you can log in any time at **www.pearsonglobaleditions.com/Tanenbaum** by providing your Login Name and Password when prompted.

*Important: The access code can only be used once. This subscription is valid for twelve months upon activation and is not transferable. If this access code has already been revealed, it may no longer be valid. If this is the case, you can purchase a subscription by going to **www.pearsonglobaleditions.com/Tanenbaum** and following the on-screen instructions.

Modern Operating Systems, Global Edition

Table of Contents

Cover

Title

Content

PREFACE

1 INTRODUCTION

1.1 WHAT IS AN OPERATING SYSTEM?

1.1.1 The Operating System as an Extended Machine

1.1.2 The Operating System as a Resource Manager

1.2 HISTORY OF OPERATING SYSTEMS

1.2.1 The First Generation (1945-55): Vacuum Tubes

1.2.2 The Second Generation (1955-65): Transistors and Batch Systems

1.2.3 The Third Generation (1965-1980): ICs and Multiprogramming

1.2.4 The Fourth Generation (1980-Present): Personal Computers

1.2.5 The Fifth Generation (1990-Present): Mobile Computers

1.3 COMPUTER HARDWARE REVIEW

1.3.1 Processors

1.3.2 Memory

1.3.3 Disks

1.3.4 I/O Devices

1.3.5 Buses

1.3.6 Booting the Computer

1.4 THE OPERATING SYSTEM ZOO

1.4.1 Mainframe Operating Systems

1.4.2 Server Operating Systems

1.4.3 Multiprocessor Operating Systems

1.4.4 Personal Computer Operating Systems

1.4.5 Handheld Computer Operating Systems

Table of Contents

- 1.4.6 Embedded Operating Systems
- 1.4.7 Sensor-Node Operating Systems
- 1.4.8 Real-Time Operating Systems
- 1.4.9 Smart Card Operating Systems

1.5 OPERATING SYSTEM CONCEPTS

- 1.5.1 Processes
- 1.5.2 Address Spaces
- 1.5.3 Files
- 1.5.4 Input/Output
- 1.5.5 Protection
- 1.5.6 The Shell
- 1.5.7 Ontogeny Recapitulates Phylogeny

1.6 SYSTEM CALLS

- 1.6.1 System Calls for Process Management
- 1.6.2 System Calls for File Management
- 1.6.3 System Calls for Directory Management
- 1.6.4 Miscellaneous System Calls
- 1.6.5 The Windows Win32 API

1.7 OPERATING SYSTEM STRUCTURE

- 1.7.1 Monolithic Systems
- 1.7.2 Layered Systems
- 1.7.3 Microkernels
- 1.7.4 Client-Server Model
- 1.7.5 Virtual Machines

1.8 THE WORLD ACCORDING TO C

- 1.8.1 The C Language
- 1.8.2 Header Files
- 1.8.3 Large Programming Projects
- 1.8.4 The Model of Run Time

1.9 RESEARCH ON OPERATING SYSTEMS

1.10 OUTLINE OF THE REST OF THIS BOOK

1.11 METRIC UNITS

Table of Contents

1.12 SUMMARY

2 PROCESSES AND THREADS

2.1 PROCESSES

- 2.1.1 The Process Model
- 2.1.2 Process Creation
- 2.1.3 Process Termination
- 2.1.4 Process Hierarchies
- 2.1.5 Process States
- 2.1.6 Implementation of Processes
- 2.1.7 Modeling Multiprogramming

2.2 THREADS

- 2.2.1 Thread Usage
- 2.2.2 The Classical Thread Model
- 2.2.3 POSIX Threads
- 2.2.4 Implementing Threads in User Space
- 2.2.5 Implementing Threads in the Kernel
- 2.2.6 Hybrid Implementations
- 2.2.7 Scheduler Activations
- 2.2.8 Pop-Up Threads
- 2.2.9 Making Single-Threaded Code Multithreaded

2.3 INTERPROCESS COMMUNICATION

- 2.3.1 Race Conditions
- 2.3.2 Critical Regions
- 2.3.3 Mutual Exclusion with Busy Waiting
- 2.3.4 Sleep and Wakeup
- 2.3.5 Semaphores
- 2.3.6 Mutexes
- 2.3.7 Monitors
- 2.3.8 Message Passing
- 2.3.9 Barriers
- 2.3.10 Avoiding Locks: Read-Copy-Update

2.4 SCHEDULING

- 2.4.1 Introduction to Scheduling

Table of Contents

- 2.4.2 Scheduling in Batch Systems
- 2.4.3 Scheduling in Interactive Systems
- 2.4.4 Scheduling in Real-Time Systems
- 2.4.5 Policy Versus Mechanism
- 2.4.6 Thread Scheduling

2.5 CLASSICAL IPC PROBLEMS

- 2.5.1 The Dining Philosophers Problem
- 2.5.2 The Readers and Writers Problem

2.6 RESEARCH ON PROCESSES AND THREADS

2.7 SUMMARY

3 MEMORY MANAGEMENT

3.1 NO MEMORY ABSTRACTION

3.2 A MEMORY ABSTRACTION: ADDRESS SPACES

- 3.2.1 The Notion of an Address Space
- 3.2.2 Swapping
- 3.2.3 Managing Free Memory

3.3 VIRTUAL MEMORY

- 3.3.1 Paging
- 3.3.2 Page Tables
- 3.3.3 Speeding Up Paging
- 3.3.4 Page Tables for Large Memories

3.4 PAGE REPLACEMENT ALGORITHMS

- 3.4.1 The Optimal Page Replacement Algorithm
- 3.4.2 The Not Recently Used Page Replacement Algorithm
- 3.4.3 The First-In, First-Out (FIFO) Page Replacement Algorithm
- 3.4.4 The Second-Chance Page Replacement Algorithm
- 3.4.5 The Clock Page Replacement Algorithm
- 3.4.6 The Least Recently Used (LRU) Page Replacement Algorithm
- 3.4.7 Simulating LRU in Software
- 3.4.8 The Working Set Page Replacement Algorithm
- 3.4.9 The WSClock Page Replacement Algorithm
- 3.4.10 Summary of Page Replacement Algorithms

Table of Contents

3.5 DESIGN ISSUES FOR PAGING SYSTEMS

- 3.5.1 Local versus Global Allocation Policies
- 3.5.2 Load Control
- 3.5.3 Page Size
- 3.5.4 Separate Instruction and Data Spaces
- 3.5.5 Shared Pages
- 3.5.6 Shared Libraries
- 3.5.7 Mapped Files
- 3.5.8 Cleaning Policy
- 3.5.9 Virtual Memory Interface

3.6 IMPLEMENTATION ISSUES

- 3.6.1 Operating System Involvement with Paging
- 3.6.2 Page Fault Handling
- 3.6.3 Instruction Backup
- 3.6.4 Locking Pages in Memory
- 3.6.5 Backing Store
- 3.6.6 Separation of Policy and Mechanism

3.7 SEGMENTATION

- 3.7.1 Implementation of Pure Segmentation
- 3.7.2 Segmentation with Paging: MULTICS
- 3.7.3 Segmentation with Paging: The Intel x86

3.8 RESEARCH ON MEMORY MANAGEMENT

3.9 SUMMARY

4 FILE SYSTEMS

4.1 FILES

- 4.1.1 File Naming
- 4.1.2 File Structure
- 4.1.3 File Types
- 4.1.4 File Access
- 4.1.5 File Attributes
- 4.1.6 File Operations
- 4.1.7 An Example Program Using File-System Calls

Table of Contents

4.2 DIRECTORIES

- 4.2.1 Single-Level Directory Systems
- 4.2.2 Hierarchical Directory Systems
- 4.2.3 Path Names
- 4.2.4 Directory Operations

4.3 FILE-SYSTEM IMPLEMENTATION

- 4.3.1 File-System Layout
- 4.3.2 Implementing Files
- 4.3.3 Implementing Directories
- 4.3.4 Shared Files
- 4.3.5 Log-Structured File Systems
- 4.3.6 Journaling File Systems
- 4.3.7 Virtual File Systems

4.4 FILE-SYSTEM MANAGEMENT AND OPTIMIZATION

- 4.4.1 Disk-Space Management
- 4.4.2 File-System Backups
- 4.4.3 File-System Consistency
- 4.4.4 File-System Performance
- 4.4.5 Defragmenting Disks

4.5 EXAMPLE FILE SYSTEMS

- 4.5.1 The MS-DOS File System
- 4.5.2 The UNIX V7 File System
- 4.5.3 CD-ROM File Systems

4.6 RESEARCH ON FILE SYSTEMS

4.7 SUMMARY

5 INPUT/OUTPUT

5.1 PRINCIPLES OF I/O HARDWARE

- 5.1.1 I/O Devices
- 5.1.2 Device Controllers
- 5.1.3 Memory-Mapped I/O
- 5.1.4 Direct Memory Access
- 5.1.5 Interrupts Revisited

Table of Contents

5.2 PRINCIPLES OF I/O SOFTWARE

5.2.1 Goals of the I/O Software

5.2.2 Programmed I/O

5.2.3 Interrupt-Driven I/O

5.2.4 I/O Using DMA

5.3 I/O SOFTWARE LAYERS

5.3.1 Interrupt Handlers

5.3.2 Device Drivers

5.3.3 Device-Independent I/O Software

5.3.4 User-Space I/O Software

5.4 DISKS

5.4.1 Disk Hardware

5.4.2 Disk Formatting

5.4.3 Disk Arm Scheduling Algorithms

5.4.4 Error Handling

5.4.5 Stable Storage

5.5 CLOCKS

5.5.1 Clock Hardware

5.5.2 Clock Software

5.5.3 Soft Timers

5.6 USER INTERFACES: KEYBOARD, MOUSE, MONITOR

5.6.1 Input Software

5.6.2 Output Software

5.7 THIN CLIENTS

5.8 POWER MANAGEMENT

5.8.1 Hardware Issues

5.8.2 Operating System Issues

5.8.3 Application Program Issues

5.9 RESEARCH ON INPUT/OUTPUT

5.10 SUMMARY

6 DEADLOCKS

6.1 RESOURCES

Table of Contents

6.1.1 Preemptable and Nonpreemptable Resources

6.1.2 Resource Acquisition

6.2 INTRODUCTION TO DEADLOCKS

6.2.1 Conditions for Resource Deadlocks

6.2.2 Deadlock Modeling

6.3 THE OSTRICH ALGORITHM

6.4 DEADLOCK DETECTION AND RECOVERY

6.4.1 Deadlock Detection with One Resource of Each Type

6.4.2 Deadlock Detection with Multiple Resources of Each Type

6.4.3 Recovery from Deadlock

6.5 DEADLOCK AVOIDANCE

6.5.1 Resource Trajectories

6.5.2 Safe and Unsafe States

6.5.3 The Bankers Algorithm for a Single Resource

6.5.4 The Bankers Algorithm for Multiple Resources

6.6 DEADLOCK PREVENTION

6.6.1 Attacking the Mutual-Exclusion Condition

6.6.2 Attacking the Hold-and-Wait Condition

6.6.3 Attacking the No-Preemption Condition

6.6.4 Attacking the Circular Wait Condition

6.7 OTHER ISSUES

6.7.1 Two-Phase Locking

6.7.2 Communication Deadlocks

6.7.3 Livelock

6.7.4 Starvation

6.8 RESEARCH ON DEADLOCKS

6.9 SUMMARY

7 VIRTUALIZATION AND THE CLOUD

7.1 HISTORY

7.2 REQUIREMENTS FOR VIRTUALIZATION

7.3 TYPE 1 AND TYPE 2 HYPERVISORS

7.4 TECHNIQUES FOR EFFICIENT VIRTUALIZATION

Table of Contents

7.4.1 Virtualizing the Unvirtualizable

7.4.2 The Cost of Virtualization

7.5 ARE HYPERVISORS MICROKERNELS DONE RIGHT?

7.6 MEMORY VIRTUALIZATION

7.7 I/O VIRTUALIZATION

7.8 VIRTUAL APPLIANCES

7.9 VIRTUAL MACHINES ON MULTICORE CPUS

7.10 LICENSING ISSUES

7.11 CLOUDS

7.11.1 Clouds as a Service

7.11.2 Virtual Machine Migration

7.11.3 Checkpointing

7.12 CASE STUDY: VMWARE

7.12.1 The Early History of VMware

7.12.2 VMware Workstation

7.12.3 Challenges in Bringing Virtualization to the x86

7.12.4 VMware Workstation: Solution Overview

7.12.5 The Evolution of VMware Workstation

7.12.6 ESX Server: VMwares type 1 Hypervisor

7.13 RESEARCH ON VIRTUALIZATION AND THE CLOUD

8 MULTIPLE PROCESSOR SYSTEMS

8.1 MULTIPROCESSORS

8.1.1 Multiprocessor Hardware

8.1.2 Multiprocessor Operating System Types

8.1.3 Multiprocessor Synchronization

8.1.4 Multiprocessor Scheduling

8.2 MULTICOMPUTERS

8.2.1 Multicomputer Hardware

8.2.2 Low-Level Communication Software

8.2.3 User-Level Communication Software

8.2.4 Remote Procedure Call

8.2.5 Distributed Shared Memory

Table of Contents

8.2.6 Multicomputer Scheduling

8.3 DISTRIBUTED SYSTEMS

8.3.1 Network Hardware

8.3.2 Network Services and Protocols

8.3.3 Document-Based Middleware

8.3.4 File-System-Based Middleware

8.3.5 Object-Based Middleware

8.3.6 Coordination-Based Middleware

8.4 RESEARCH ON MULTIPLE PROCESSOR SYSTEMS

8.5 SUMMARY

9 SECURITY

9.1 THE SECURITY ENVIRONMENT

9.1.1 Threats

9.1.2 Attackers

9.2 OPERATING SYSTEMS SECURITY

9.2.1 Can We Build Secure Systems?

9.2.2 Trusted Computing Base

9.3 CONTROLLING ACCESS TO RESOURCES

9.3.1 Protection Domains

9.3.2 Access Control Lists

9.3.3 Capabilities

9.4 FORMAL MODELS OF SECURE SYSTEMS

9.4.1 Multilevel Security

9.4.2 Covert Channels

9.5 BASICS OF CRYPTOGRAPHY

9.5.1 Secret-Key Cryptography

9.5.2 Public-Key Cryptography

9.5.3 One-Way Functions

9.5.4 Digital Signatures

9.5.5 Trusted Platform Modules

9.6 AUTHENTICATION

9.6.1 Authentication Using a Physical Object

Table of Contents

9.6.2 Authentication Using Biometrics

9.7 EXPLOITING SOFTWARE

9.7.1 Buffer Overflow Attacks

9.7.2 Format String Attacks

9.7.3 Dangling Pointers

9.7.4 Null Pointer Dereference Attacks

9.7.5 Integer Overflow Attacks

9.7.6 Command Injection Attacks

9.7.7 Time of Check to Time of Use Attacks

9.8 INSIDER ATTA CKS

9.8.1 Logic Bombs

9.8.2 Back Doors

9.8.3 Login Spoofing

9.9 MALWARE

9.9.1 Trojan Horses

9.9.2 Viruses

9.9.3 Worms

9.9.4 Spyware

9.9.5 Rootkits

9.10 DEFENSES

9.10.1 Firewalls

9.10.2 Antivirus and Anti-Antivirus Techniques

9.10.3 Code Signing

9.10.4 Jailing

9.10.5 Model-Based Intrusion Detection

9.10.6 Encapsulating Mobile Code

9.10.7 Java Security

9.11 RESEARCH ON SECURITY

9.12 SUMMARY

10 CASE STUDY 1: UNIX, LINUX, AND ANDROID

10.1 HISTORY OF UNIX AND LINUX

10.1.1 UNICS

Table of Contents

10.1.2 PDP-11 UNIX

10.1.3 Portable UNIX

10.1.4 Berkeley UNIX

10.1.5 Standard UNIX

10.1.6 MINIX

10.1.7 Linux

10.2 OVERVIEW OF LINUX

10.2.1 Linux Goals

10.2.2 Interfaces to Linux

10.2.3 The Shell

10.2.4 Linux Utility Programs

10.2.5 Kernel Structure

10.3 PROCESSES IN LINUX

10.3.1 Fundamental Concepts

10.3.2 Process-Management System Calls in Linux

10.3.3 Implementation of Processes and Threads in Linux

10.3.4 Scheduling in Linux

10.3.5 Booting Linux

10.4 MEMORY MANAGEMENT IN LINUX

10.4.1 Fundamental Concepts

10.4.2 Memory Management System Calls in Linux

10.4.3 Implementation of Memory Management in Linux

10.4.4 Paging in Linux

10.5 INPUT/OUTPUT IN LINUX

10.5.1 Fundamental Concepts

10.5.2 Networking

10.5.3 Input/Output System Calls in Linux

10.5.4 Implementation of Input/Output in Linux

10.5.5 Modules in Linux

10.6 THE LINUX FILE SYSTEM

10.6.1 Fundamental Concepts

10.6.2 File-System Calls in Linux

10.6.3 Implementation of the Linux File System

Table of Contents

10.6.4 NFS: The Network File System

10.7 SECURITY IN LINUX

10.7.1 Fundamental Concepts

10.7.2 Security System Calls in Linux

10.7.3 Implementation of Security in Linux

10.8 ANDROID

10.8.1 Android and Google

10.8.2 History of Android

10.8.3 Design Goals

10.8.4 Android Architecture

10.8.5 Linux Extensions

10.8.6 Dalvik

10.8.7 Binder IPC

10.8.8 Android Applications

10.8.9 Intents

10.8.10 Application Sandboxes

10.8.11 Security

10.8.12 Process Model

10.9 SUMMARY

11 CASE STUDY 2: WINDOWS 8

11.1 HISTORY OF WINDOWS THROUGH WINDOWS 8.1

11.1.1 1980s: MS-DOS

11.1.2 1990s: MS-DOS-based Windows

11.1.3 2000s: NT-based Windows

11.1.4 Windows Vista

11.1.5 2010s: Modern Windows

11.2 PROGRAMMING WINDOWS

11.2.1 The Native NT Application Programming Interface

11.2.2 The Win32 Application Programming Interface

11.2.3 The Windows Registry

11.3 SYSTEM STRUCTURE

11.3.1 Operating System Structure

Table of Contents

11.3.2 Booting Windows

11.3.3 Implementation of the Object Manager

11.3.4 Subsystems, DLLs, and User-Mode Services

11.4 PROCESSES AND THREADS IN WINDOWS

11.4.1 Fundamental Concepts

11.4.2 Job, Process, Thread, and Fiber Management API Calls

11.4.3 Implementation of Processes and Threads

11.5 MEMORY MANAGEMENT

11.5.1 Fundamental Concepts

11.5.2 Memory-Management System Calls

11.5.3 Implementation of Memory Management

11.6 CACHING IN WINDOWS

11.7 INPUT/OUTPUT IN WINDOWS

11.7.1 Fundamental Concepts

11.7.2 Input/Output API Calls

11.7.3 Implementation of I/O

11.8 THE WINDOWS NT FILE SYSTEM

11.8.1 Fundamental Concepts

11.8.2 Implementation of the NT File System

11.9 WINDOWS POWER MANAGEMENT

11.10 SECURITY IN WINDOWS 8

11.10.1 Fundamental Concepts

11.10.2 Security API Calls

11.10.3 Implementation of Security

11.10.4 Security Mitigations

11.11 SUMMARY

12 OPERATING SYSTEM DESIGN

12.1 THE NATURE OF THE DESIGN PROBLEM

12.1.1 Goals

12.1.2 Why Is It Hard to Design an Operating System?

12.2 INTERFACE DESIGN

12.2.1 Guiding Principles

Table of Contents

12.2.2 Paradigms

12.2.3 The System-Call Interface

12.3 IMPLEMENTATION

12.3.1 System Structure

12.3.2 Mechanism vs. Policy

12.3.3 Orthogonality

12.3.4 Naming

12.3.5 Binding Time

12.3.6 Static vs. Dynamic Structures

12.3.7 Top-Down vs. Bottom-Up Implementation

12.3.8 Synchronous vs. Asynchronous Communication

12.3.9 Useful Techniques

12.4 PERFORMANCE

12.4.1 Why Are Operating Systems Slow?

12.4.2 What Should Be Optimized?

12.4.3 Space-Time Trade-offs

12.4.4 Caching

12.4.5 Hints

12.4.6 Exploiting Locality

12.4.7 Optimize the Common Case

12.5 PROJECT MANAGEMENT

12.5.1 The Mythical Man Month

12.5.2 Team Structure

12.5.3 The Role of Experience

12.5.4 No Silver Bullet

12.6 TRENDS IN OPERATING SYSTEM DESIGN

12.6.1 Virtualization and the Cloud

12.6.2 Manycore Chips

12.6.3 Large-Address-Space Operating Systems

12.6.4 Seamless Data Access

12.6.5 Battery-Powered Computers

12.6.6 Embedded Systems

12.7 SUMMARY

Table of Contents

13 READING LIST AND BIBLIOGRAPHY

13.1 SUGGESTIONS FOR FURTHER READING

- 13.1.1 Introduction
- 13.1.2 Processes and Threads
- 13.1.3 Memory Management
- 13.1.4 File Systems
- 13.1.5 Input/Output
- 13.1.6 Deadlocks
- 13.1.7 Virtualization and the Cloud
- 13.1.8 Multiple Processor Systems
- 13.1.9 Security
- 13.1.10 Case Study 1: UNIX, Linux, and Android
- 13.1.11 Case Study 2: Windows 8
- 13.1.12 Operating System Design

13.2 ALPHABETICAL BIBLIOGRAPHY

INDEX