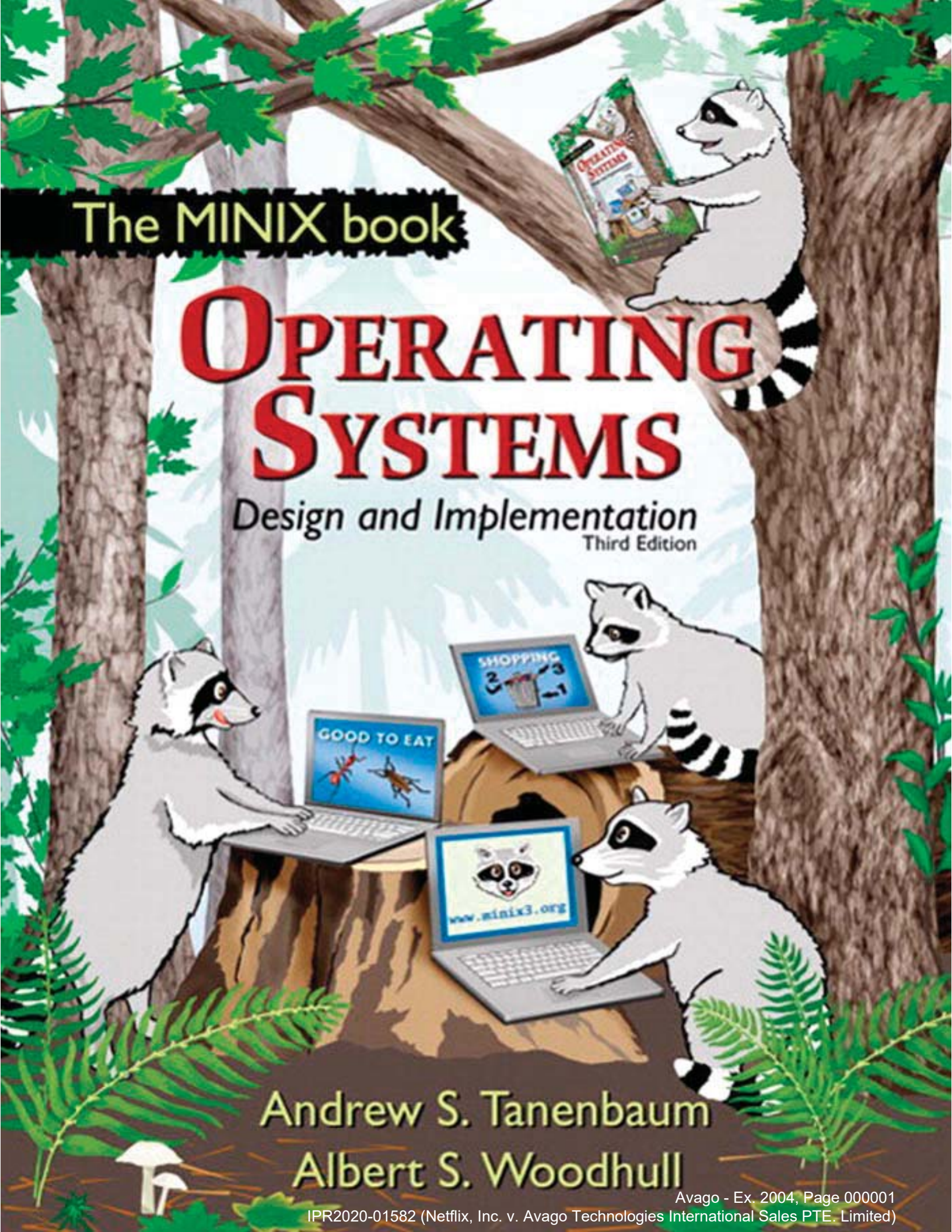

OPERATING SYSTEMS DESIGN AND IMPLEMENTATION

Third Edition

This page intentionally left blank

OPERATING SYSTEMS DESIGN AND IMPLEMENTATION

Third Edition

ANDREW S. TANENBAUM
Vrije Universiteit
Amsterdam, The Netherlands

ALBERT S. WOODHULL
Amherst, Massachusetts

Upper Saddle River, New Jersey 07458

Library of Congress Cataloging in Publication Data
Tanenbaum, Andrew S.
Operating Systems: Design and Implementation / Andrew S. Tanenbaum, Albert S. Woodhull. -- 3rd ed.
ISBN: 0-13-142938-8
1. Operating systems (Computers) I. Woodhull, Albert S. II. Title

QA76.76.O63T36 2006
005.4'3--dc22

Vice President and Editorial Director, ECS: *Marcia J. Horton*
Executive Editor: *Tracy Dunkelberger*
Editorial Assistant: *Christianna Lee*
Executive Managing Editor: *Vince O'Brien*
Managing Editor: *Camille Trentacoste*
Director of Creative Services: *Paul Belfanti*
Art Director and Cover Manager: *Heather Scott*
Cover Design and Illutsration: *Tamara Newnam*
Managing Editor, AV Management and Production: *Patricia Burns*
Art Editor: *Gregory Dulles*
Manufacturing Manager, ESM: *Alexis Heydt-Long*
Manufacturing Buyer: *Lisa McDowell*
Executive Marketing Manager: *Robin O'Brien*
Marketing Assistant: *Barrie Reinhold*

Pearson Prentice Hall
Pearson Education, Inc.
Upper Saddle River, NJ 07458

10 9 8 7 6 5 4 3 2 1

ISBN 0-13-0-13-142938-8

Pearson Education Ltd., London
Pearson Education Australia Pty. Ltd., Sydney
Pearson Education Singapore, Pte. Ltd.
Pearson Education North Asia Ltd., Hong Kong
Pearson Education Canada, Inc., Toronto
Pearson Educación de Mexico, S.A. de C.V.
Pearson Education-Japan, Tokyo
Pearson Education Malaysia, Pte. Ltd.
Pearson Education, Inc., Upper Saddle River, New Jersey

To Suzanne, Barbara, Marvin, and the memory of Sweetie π and Bram

- AST

To Barbara and Gordon

- ASW

The MINIX 3 Mascot

Other operating systems have an animal mascot, so we felt MINIX 3 ought to have one too. We chose the raccoon because raccoons are small, cute, clever, agile, eat bugs, and are user-friendly—at least if you keep your garbage can well locked.

This page intentionally left blank

CONTENTS

1

INTRODUCTION

Without its software, a computer is basically a useless lump of metal. With its software, a computer can store, process, and retrieve information; play music and videos; send e-mail, search the Internet; and engage in many other valuable activities to earn its keep. Computer software can be divided roughly into two kinds: system programs, which manage the operation of the computer itself, and application programs, which perform the actual work the user wants. The most fundamental system program is the **operating system**, whose job is to control all the computer's resources and provide a base upon which the application programs can be written. Operating systems are the topic of this book. In particular, an operating system called MINIX 3 is used as a model, to illustrate design principles and the realities of implementing a design.

A modern computer system consists of one or more processors, some main memory, disks, printers, a keyboard, a display, network interfaces, and other input/output devices. All in all, a complex system. Writing programs that keep track of all these components and use them correctly, let alone optimally, is an extremely difficult job. If every programmer had to be concerned with how disk drives work, and with all the dozens of things that could go wrong when reading a disk block, it is unlikely that many programs could be written at all.

Many years ago it became abundantly clear that some way had to be found to shield programmers from the complexity of the hardware. The way that has evolved gradually is to put a layer of software on top of the bare hardware, to manage all parts of the system, and present the user with an interface or **virtual**

machine that is easier to understand and program. This layer of software is the operating system.

The placement of the operating system is shown in Fig. 1-1. At the bottom is the hardware, which, in many cases, is itself composed of two or more levels (or layers). The lowest level contains physical devices, consisting of integrated circuit chips, wires, power supplies, cathode ray tubes, and similar physical devices. How these are constructed and how they work is the province of the electrical engineer.

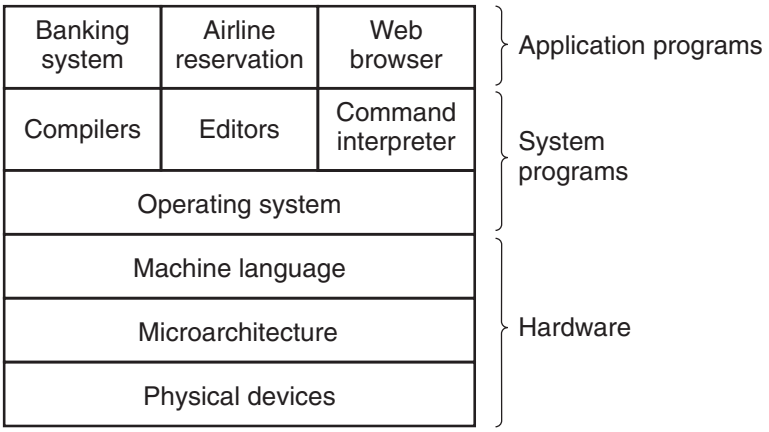

Figure 1-1. A computer system consists of hardware, system programs, and application programs.

Next comes the **microarchitecture level**, in which the physical devices are grouped together to form functional units. Typically this level contains some registers internal to the CPU (Central Processing Unit) and a data path containing an arithmetic logic unit. In each clock cycle, one or two operands are fetched from the registers and combined in the arithmetic logic unit (for example, by addition or Boolean AND). The result is stored in one or more registers. On some machines, the operation of the data path is controlled by software, called the **microprogram**. On other machines, it is controlled directly by hardware circuits.

The purpose of the data path is to execute some set of instructions. Some of these can be carried out in one data path cycle; others may require multiple data path cycles. These instructions may use registers or other hardware facilities. Together, the hardware and instructions visible to an assembly language programmer form the **ISA** (**Instruction Set Architecture**) This level is often called **machine language**.

The machine language typically has between 50 and 300 instructions, mostly for moving data around the machine, doing arithmetic, and comparing values. In this level, the input/output devices are controlled by loading values into special **device registers**. For example, a disk can be commanded to read by loading the values of the disk address, main memory address, byte count, and direction (read or write) into its registers. In practice, many more parameters are needed, and the