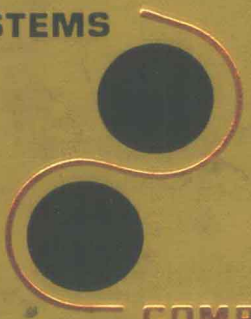
OPERATING SYSTEMS



COMPUTER
SCIENCE
SERIES

Operating Systems

Stuart E. Madnick
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ALFRED P. SLOAN SCHOOL OF MANAGEMENT
and Project MAC

John J. Donovan
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ALFRED P. SLOAN SCHOOL OF MANAGEMENT*
and Project MAC

*Formerly with the Department of Electrical Engineering, M.I.T.

McGraw-Hill, Inc.
New York St. Louis San Francisco Auckland Bogotá
Caracas Lisbon London Madrid Mexico City Milan
Montreal New Delhi San Juan Singapore
Sydney Tokyo Toronto

OPERATING SYSTEMS

22 23 24 25 26 27 BKMBKM 9 9 8 7 6 5 4

Library of Congress Cataloging in Publication Data

Madnick, Stuart E
Operating systems.

(McGraw-Hill computer science series)
Bibliography: p.
1. Electronic digital computers—Programming.
I. Donovan, John J., joint author. II. Title.
QA76.6.M33 001.6'42 74-2040
ISBN 0-07-039455-5

This book was set in Press Roman by Allen Wayne Technical Corp. The
editor was Kenneth J. Bowman; the designer was Allen Wayne Technical
Corp.; the production supervisor was Bill Greenwood.

To all those who have supported me when I most needed encouragement or help, from my high school teacher, Mr. Thornton; my college advisors, Professors Shan Kuo and Conrad Wogrin; my wife, Marilyn; my co-author, friend, and colleague Stuart Madnick; Professor Licklider; my friend James Nichols; to my M.I.T. friend and critic, Professor Paul MacAvoy.

DONOVAN

To my wife, Ethel, and my children, Howard, Michael, and Lynne.

MADNICK

Preface

Purpose of Book

This book presents advanced software techniques, especially focusing on operating systems. It presents material that will enable the reader to design, use, and analyze not only current operating systems but future ones as well. The presentation covers a spectrum ranging from specific implementation details to relevant theoretical results. The book is written as a text, with relevant problems, examples, and exercises.

Approach of Book

Our approach is to develop a framework based upon the concept of resource management. Within this framework, specific case studies are introduced as examples and relevant theory is presented where appropriate.

The book is unique in that it presents a comprehensive framework for the design, study, and implementation of operating systems. Most other texts either present miscellaneous theoretical results or use a case study approach. In the first case, the student may not learn where or how to apply the theoretical results. In the second, he may find it difficult to differentiate between fundamental concepts and design issues and arbitary decisions.

The features we attempt to provide in this book are the following:

1 It is a package course. This is not just a collection of articles but an integrated textbook.

2 It is a text for upper undergraduate or graduate courses in computer science or management.

3 Questions appear at the end of each chapter.

4 A teacher's manual is available with solutions to the questions and possible course syllabuses.

5 The book is based on a class-tested course given at M.I.T. for several years. It has been used at other universities, such as the University of Rhode Island and the University of Pittsburgh, and at several industrial companies, for example, The Foxboro Company, U.S. Naval Underwater Systems Center.

6 The text relates to existing operating systems. We draw examples particularly, though not exclusively, from IBM's System/370.

7 The book uses a framework approach that our students find conceptually enjoyable and understandable.

8 Emphasis is given to relevant material. We have found that most students cannot appreciate theoretical results without some practical motivation.

9 A complete sample operating system is presented. Studying an operating system is similar to riding a bicycle—you can write the equations to show the bicycle will balance but you can never ride the bicycle until you actually try.

10 A comprehensive and annotated bibliography is provided to enable the reader to delve further into the topics introduced in the text.

11 Teaching aids are available. In addition to this book, we have made the following available through the M.I.T. computation center:
   a A 360 machine simulator written in PL/I.
   b Grading programs for simple assembly language problems.
   c Sample PL/I problems and a facility for running them.
   d Sample I/O and interrupt problems.
   e An environment simulator for testing and running different process schedulers (dispatchers).
   f An I/O environment simulator for testing and evaluating different I/O scheduling algorithms.
   g A fast PL/I compiler—this uses the same source as IBM's PL/I(F) and produces the same object code, but in less time.
   h A submonitor system for running student jobs.

All the programs listed in 11 above are described in our book, ***Software Projects: Pedagogical Aids for Software Education and Research*** (see page 581), which describes all the programs listed in 11 above. Specifically, the description for each program consists of three sections:

1 A student guide—sample of student handouts for the package.

2 A teacher's guide—descriptions of how a teacher would operate the program.

3 A maintenance manual—a complete description of the implementation of the program.

Uses of Book

As noted above, this book is primarily a text on an upper undergraduate or graduate level. As a prerequisite to the material presented, the student should be familiar with the concepts of an assembler, loader, compiler, and should have programming experience in some high-level language. Assembly language programming is desirable. A good background can be found in the text ***Systems Programming*** by John J. Donovan.

We see four ways in which this book may be used as a text:

1 As an intensive one-semester course in operating systems.

2 As a two-term sequence:
   a First term: course introducing the concepts and practice of I/O programming and interrupt processing. The course would provide the conceptual frame-

work necessary for further study in operating systems as well as the practical experience for programming real-time systems, minicomputers, and specialized applications (e.g., patient-monitoring computers).

b Second term: high-level course focusing on the concepts, design, implementation, and relevant theory of operating systems.

3 To formulate courses using individual sections of the book, e.g.,

a Chapter 2—a course devoted to applications of real-time programming, I/O programming, and interrupts.

b Chapter 7—a workshop or lab course constructing sample operating systems.

c An overview course on operating systems, skipping the details of Chapter 2 and focusing on the theory and concepts of Chapters 3, 4, 5, and 6.

4 As a course for professionals or for self-study using the problems and solutions manuals.

The book is aimed at helping the student in the following areas:

a Designing operating systems

b Understanding relevant theory

c Using techniques presented in operating systems in other applications

d Using operating systems

e Buying or evaluating operating systems

As a text, the book covers the topics mentioned in the recent study by the ACM COSINE Committee, "An undergraduate course in operating systems." The text further covers material contained in two courses of Curriculum 68 as described by the ACM Curriculum Committee in Computer Science[1]: (A2) Advanced Computer Organization and (A8) Large Scale Information Systems. The book covers material as reported by ACM Curriculum Committee on Computer Education for Management focusing on Curriculum for Graduate Professional Programs in Information Systems.[2] These are (C10) Introduction to Computer Systems and Programming; (C11) Information Structure and Files (Chapter 6 of this book); (C2) Computer Systems; (D2) System Design; and (D3) Systems Development Projects (Chapter 7).

In Chapter 2 we present the basic material of machine structures, machine language programming, I/O programming, and interrupt handling. We feel that a person knowledgeable in operating systems must not be intimidated by "pushing bits." This chapter uses the IBM System/370 for detailed examples (this is the only chapter that is heavily machine-dependent).

In Chapters 3, 4, 5, and 6 we develop the framework for the analysis and design of operating systems. In our view of an operating system, the system is a manager of resources. (We differentiate four resources: memory, processors, devices, information.) These chapters discuss the techniques and issues involved in managing these resources and develop the framework for managing them through examples, starting with the simplest and moving to the most complicated. The maximum function (taking the most complicated example of each resource) of these chapters

[1] Communications of the Association of Computing Machinery (CACM), Vol. 11, No. 3, p. 151 (March, 1968).

[2] Communications of the Association of Computing Machinery (CACM), Vol. 15, No. 5, p. 363 (May, 1972).

is a system like MULTICS, whereas the minimum function is a system such as IBM's OS/360 Primary Control Program (PCP).

In Chapter 7 we design an operating system using the framework developed in previous chapters. We start with the general strategy, develop descriptions of the databases and routines, and finally present the code for an operational implementation.

When the management of all these resources is put in one system, we note that they are not independent. For example, the processor management technique may depend upon whether or not the system uses a paged or swapping memory management technique. In Chapter 8 we explore the interrelationships among the managers of these resources and develop techniques for performance evaluation.

Chapter 9 analyzes various existing systems within the framework developed, and Chapter 10 contains an annotated bibliography of the field.

Stuart E. Madnick
John J. Donovan

Acknowledgments

We list here the people who assisted us in writing this book.

We thank Charles A. Ziering, our head teaching assistant, and particularly acknowledge his reliability. We especially note his contributions to Chapters 4 and 8, as he confirmed (and corrected) most of our figures, and his work on the solutions manual.

We thank our teaching assistants, Paul Gregory, Judy Piggins, Harry Forsdick, Jeff Bunza, Mike Knaur, and Suhundra Umarji, for contributing to the questions, David Schwartz for his work on the bibliography, Chad Carpenter and Jeff Buzen for proofreading, and Anne Beer for her assistance with the index.

Chapter 7 evolved from the thesis work of John DeTreville and was further refined by Richard Swift.

We give special note to those M.I.T. students who went through early versions of our course and survived—we thank them for their constructive criticisms.

We thank Ellen Nangle, who has once again edited, proofread, and typed a book singlehandedly.

We especially appreciate the fine editing of Irene Gunther, the perseverance of Nora Braverman, and the innovative styling of Irving Weinstock, all of Allen Wayne Technical Corp.

We thank Professors Michael Scott-Morton and John Little of the Sloan School for providing a healthy environment within the Management Science group for producing this book.

Stuart E. Madnick
John J. Donovan

Contents

Framework of Book

1

This book has two major objectives: to teach the fundamentals of advanced software engineering (including asynchronous operation, interrupt processing, operating system interfaces, and hardware-software tradeoffs), and to enable the reader to design and analyze operating systems.

Modern computer hardware is very powerful (see Figure 1-1a) and can be used for many purposes. But the hardware often provides an awkward or difficult-to-use interface to the world. In order to tame this "bare machine," we have developed operating systems that can manage the basic hardware resources and provide a more hospitable interface to users and their programs (see Figure 1-1b). Operating systems have become so essential to efficient computer operation that many people view them as inseparable from the hardware.

There are many important reasons for studying operating systems; the most notable are: (1) for special-purpose usage you may have to design your own operating system or modify an existing one; (2) the selection of the operating system and its options is a major decision for most computer installations; (3) the user must interact with the operating system in order to accomplish his task since it is his primary interface with the computer; and (4) many concepts and techniques found in operating systems have general applicability in other applications. The framework presented in this book is especially suited to addressing these issues.

The term *operating system* denotes those program modules within a computer system that govern the control of equipment resources such as processors, main storage, secondary storage, I/O devices, and files. These modules resolve conflicts, attempt to optimize performance, and simplify the effective use of the system. They act as an interface between the user's programs and the physical computer hardware.

These modules are sometimes collectively called the *Control*, the *Monitor*,[1] the *Executive*, the *Supervisor*, or the *Operating System*. Applications-oriented modules,

[1] Not to be confused with a *Monster*.

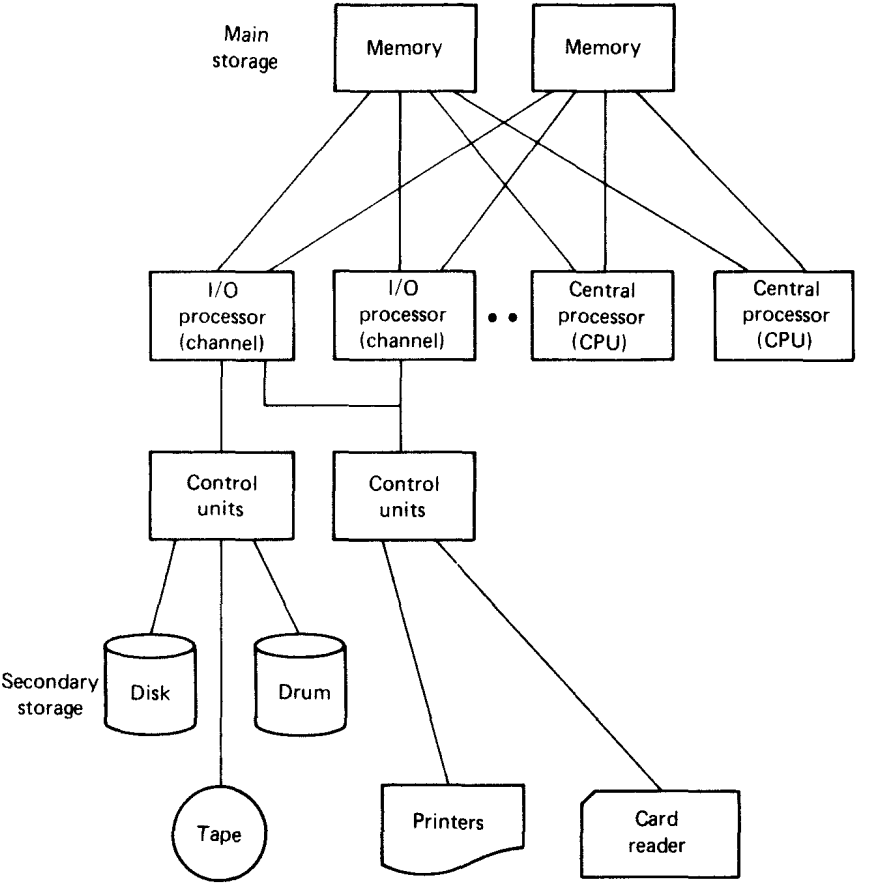


Figure 1-1a Basic computer hardware (bare machine)

language processors, library routines, and debugging aids are not included in our definition of an operating system. These are merely users of the operating system and are covered adequately elsewhere (Donovan, 1972; Knuth, 1968).

1-1 IMPORTANCE OF OPERATING SYSTEMS

Computers have become essential to society in such traditional applications as payroll and banking where the volume of operations performed makes them indispensable. For example, in the mid-1950s the Bank of America pioneered the use of computers in banking after realizing that, in the foreseeable future, it would require the entire adult population of California to handle checks manually (Fano, 1972). Computers are now being used in new fields, such as the design of automobiles, improved medical care, and space exploration; future applications are being envisaged in the areas of learning, acquiring knowledge, and artificial intelligence.