Operating Systems

A Systematic View

Davis

OPERATING SYSTEMS

A Systematic View

WILLIAM S. DAVIS, *Miami University*

ADDISON-WESLEY PUBLISHING COMPANY

Reading, Massachusetts
Menlo Park, California · London · Amsterdam · Don Mills, Ontario · Sydney

ISBN 0-201-01118-2
BCDEFGHIJ-HA-798

PREFACE

In the summer of 1968, I took my first course in operating systems. The course was taught by an IBM (my employer at the time) programmer. The class was composed of "new hires" and "transfers," all extremely interested in demonstrating their skills to a new employer; in educational parlance, we'd say they were well motivated.

We began by digging into the actual code for the SVC interrupt handler and moved up from there. It didn't work. We were all "lost" right from the start. When intelligent, well-motivated students don't learn, something is wrong. A "bit-level" discussion of actual operating system code is *not* the way to introduce a student to operating systems.

Working as a systems analyst and programmer over the next three years, I gradually, through a process of "osmosis," began to learn about operating systems. As a result of a fortunate series of job assignments, I became deeply involved with two very different operating systems: one, a developing special-purpose production control system based on DOS, and the other, a general-purpose OS/MVT system. A comparison of similarities and differences contributed greatly to my education. Things really began to "click" into place once I had formulated a good, general overview of how all the diverse pieces of a given operating system fit together. To put it another way, I began to understand operating systems only after learning to view them as operating *systems* rather than as groups of diverse, independent functions.

After leaving IBM and resuming my teaching career at Miami University, I was asked to teach a course in Operating Systems, thus beginning a search for a textbook. Most good Operating System texts are aimed at the advanced undergraduate or graduate computer science major. Their ap-

proach tends to be quite mathematical; they assume *substantial* prior exposure to both computer concepts and higher mathematics. I was to teach sophomores.

Are sophomore-level operating system courses unusual? I think not. Our approach, in Miami's Department of Systems Analysis, is to give our students a solid introduction to the *use* of computers during their first two years, branching into such topics as: mathematical modeling, operations research, systems analysis and design, and courses in related disciplines during the junior and senior years. In addition to our core of majors, we have a large number of actual and "de facto" minors interested (for employment-related reasons) in our freshman and sophomore sequence; most majors *and* minors take Operating Systems. In addition, we have two Associate Degree programs, both incorporating the Operating System course. Other schools offer similar programs; a knowledge of computer programming and the use of computers is viewed as a valuable "marketable skill" in today's job market.

One quarter of trying to teach operating system theory was enough to convince me that a mathematical approach does not work at the sophomore or minor level; the comments and input of my colleagues merely reinforced my view. My own experience told me that a "code it" approach doesn't work either. What was needed, I felt, was a functional-level approach, showing the student the major components of an operating system and illustrating how these pieces fit together to form an operating *system*. I've tried it, and it works! Given the results of my classroom experiments and my view of the potential market, I decided to write this book.

My approach is most definitely *not* theoretical; my intent is to show *why* operating systems are needed and *what*, at a functional, "macro" level, they do. This book does not get into a bit-level discussion of the actual implementation of operating system modules on a given machine. I've assumed that students using this book have completed at least one full year of computer-related studies, usually including an introductory course and one or more programming courses; assembler language exposure would be an ideal, but it's not essential. A professional programmer working with almost any language should have very little trouble with this material. Almost no background in theoretical mathematics has been assumed.

The book has been divided into five sections. Any operating system, at its core, is a resource manager; thus Part I discusses the basic resources of any computer system: hardware, software, and data. My intent in this section is to present basic concepts as they relate to operating system development and to fill in any gaps in the prior preparation of a given student (thus the few pages on number systems for the benefit of the compiler level programmer). To most students, part (perhaps even all) of this material will be review in nature.

In Part II, we begin to move into the development of operating systems, stressing why they are needed and, at a very general level, what functions are performed. Such topics as: compilers, libraries, access methods, multiprogramming, spooling, and others are introduced in this section.

Job Control Language is introduced in Part III, with Chapter 7 covering the basics of IBM's DOS job control and Chapters 8 and 9 getting into S job control. My intent is to show the need for a formal means of job control on a modern, multiprogrammed system; any job control language would do equally well, so instructors should feel free to substitute the job control language of an "on-site" non-IBM computer. This material is inserted near the beginning of the book to allow adequate time for the completion of student exercises. Part III can be skipped, totally or in part, without losing text continuity.

In Part IV we begin to dig more deeply into actual operating systems, describing the operating environment of the IBM System/360 and System/370 series of computers (the PSW, interrupts, and channel communications) and showing how, at a macro level, two different operating systems, DOS and OS/MFT, are implemented in this environment. IBM products were chosen as examples simply because IBM is the dominant supplier of computers, meaning that more students will have access to IBM equipment than to the equipment of any other manufacturer. After covering these two systems in some depth, we move on to the general question of memory management, covering such topics as: variable length regions, dynamic memory relocation, program segmentation and paging, virtual memory, and memory hierarchies.

In the final section, we leave general-purpose operating systems and move into a discussion of a number of special-purpose systems. The primary objective of this section is to illustrate that an operating system, to be considered effective, must meet the objectives of a given application. In this section, we also spend some time on the "hardware vs. software" question, indicating that many operating system functions can be implemented through either hardware or software, or a combination of both.

The book is designed for second year students in a program oriented toward the *use* of computers rather than toward the design of computers. In addition to the traditional four-year school preparing majors or minors, community colleges should find this book useful. In a theoretical "computer science" program, this book might support a first course in operating systems, giving the student a framework against which to measure later, more advanced studies.

Hamilton, Ohio
October 1976

W. S. D.

ABRIDGED CONTENTS

Chapter 4 Data Management and File Structures

PART II OPERATING SYSTEM DEVELOPMENT

Chapter 5 Single Program Systems: the Second Generation

Chapter 6 Multiprogramming and Time-Sharing

CHAPTER 1

Introduction and Overview

The purpose of any data-processing system is to convert data into more useful information; i.e., to process data. An electronic data-processing system combines hardware, software, and data resources toward meeting this objective. These resources are expensive. Many firms spend millions of dollars each year on hardware—often even more on programming and data management. Because of this high cost, it is essential that these resources be used as efficiently as possible. Operating systems have been developed with this idea in mind—to improve the efficiency of a data-processing system.

Note carefully the use of the word "system." A well-designed operating system is *not* concerned with just hardware or just software or just data management, but with optimizing the way in which *all* of these resources work together in achieving some desired objective. Not all data-processing systems have the same objective—a manufacturing process-control system may stress speed of response while an educational system at a university may stress flexibility. Thus not all installations will want the same operating system; "best" is a relative term. In this text, we'll be discussing operating systems not as an end in themselves but as solutions to a number of data-processing problems, always keeping system objectives in mind.

The purpose of this book is to give the reader a basic understanding of what an operating system is and how it works. Specific examples of operating system design and implementation will be used to illustrate a number of points; we'll try to avoid the bit-level discussion of the intimate working details of the products of any one manufacturer or the theory of operating system design. Our objective is to illustrate the *problems* handled by operating systems and not any single set of solutions to these problems. The text is designed to support a first

course in operating systems. The concentration is on the application of this specialized software to a real-world environment; this is *not* a theoretical text.

The book is divided into five parts. Part I, Chapters 2 through 4, covers the basic concepts of software, hardware, and data—the system resources which are managed by an operating system. For many students, much of this material will be review in nature; it's included because subsequent chapters assume a knowledge of this information.

Part II, Chapters 5 and 6, follows the rapidly developing technology of the past two decades and the parallel evolution of operating systems. The concepts of multiprogramming and time sharing are introduced in this section. Emphasis is placed on the importance of economic factors in these developments.

In Part III, we study modern programmer/system communications by analyzing two common job control languages. In Chapter 7, job control for IBM's Disk Operating System (DOS) will be studied; Chapters 8 and 9 concentrate on the job control language for IBM's full operating system (OS/JCL), with the JOB and EXEC cards being covered in Chapter 8 and the DD card in Chapter 9. The products of IBM have been chosen for a very obvious reason—IBM is the dominant force in the computer market. Not all features of the job control languages are covered in this section, only those more commonly used. The intent is to illustrate modern programmer/system communications and not to present an exhaustive course in JCL; the beginning programmer should, however, find the application orientation of this material useful in handling many everyday programming problems.

Part IV, Chapters 10 through 15, covers a number of general-purpose operating systems. Chapter 10 summarizes the basic functions of any operating system, concentrating on various measures of system effectiveness. The next three chapters, 11 through 13, are related; their purpose is to describe a hardware environment and two different operating systems designed to work under this environment. Operating principles of the IBM System/360 (Chapter 11) and two of IBM's operating systems, DOS and OS/MFT, have been chosen to illustrate these ideas. Actually, the products of almost any computer manufacturer would have done as well; however, the products of IBM are known to more potential readers than are those of any other firm. Chapter 14 generalizes the material presented in the preceding three chapters, relating it to other manufacturers' products. The final chapter in this section, Chapter 15, introduces the key concepts of virtual memory and paging.

Finally, in Part V, Chapters 16 through 19, a number of special-purpose systems and their associated support software will be discussed. Chapter 16 will concentrate on production-control applications where the high cost of manufacturing downtime creates a need for rapid response and high reliability. In Chapter 17, data base management and data communications are discussed

in the context of a management information application. Applications involving more than one computer, primarily minicomputer and full-sized computer combinations, are discussed. In each of these chapters, we'll be considering the application in its environmental context, with emphasis on how the operating system and other support software help to maximize the utilization of system resources given the measures of effectiveness most crucial to the application.

The text has been written to support a four-credit semester course; by skipping either Part III or Part V, it could support a four-credit quarter course. For most students, much of Part I will be a review. Exposure to at least one programming language has been assumed; readers with no assembly-language background should read the chapter on software carefully, as a basic understanding of binary numbers will be important in Part IV when we get into the operating principles of the IBM System/360. The chapter on hardware relates equipment to a number of concepts we'll be discussing later in the book; channels, control units, and teleprocessing hardware may be unfamiliar. The material on data and file organizations may be new to many readers.

Part II is written in a very nontechnical manner and should provide the student, business manager, engineer, or computer professional with a good, basic understanding of what operating systems are all about. The chapters on job control might be used to introduce this topic at any level. Parts IV and V concentrate on operating systems functions, at a "macro" level, avoiding the bit-level details of operating-system design; given an understanding of the introductory material in Part I, a programmer with a background in any language should find the material quite readable.