Computer Science and Applied Mathematics

OPERATING SYSTEMS

Dionysios C. Tsichritzis
Philip A. Bernstein

OPERATING SYSTEMS

Dionysios C. Tsichritzis and Philip A. Bernstein
University of Toronto

ACADEMIC PRESS New York and London
A Subsidiary of Harcourt Brace Jovanovich, Publishers

ACADEMIC PRESS, INC.
111 Fifth Avenue, New York, New York 10003

United Kingdom Edition published by
ACADEMIC PRESS, INC. (LONDON) LTD.
24/28 Oval Road, London NW1

Library of Congress Cataloging in Publication Data

Tsichritzis, Dionysios C
Operating systems.

(Computer science and applied mathematics)
Bibliography: p.
1. Electronic digital computers. I. Bernstein, Philip A., joint author. II. Title.
QA76.5.T738 001.6'44'04 73-18959
ISBN 0-12-701750-X

PRINTED IN THE UNITED STATES OF AMERICA

PREFACE

The concept of an operating system is difficult to define precisely. The word "system" is used with a multitude of meanings in both computer science and engineering, and the word "operating" does not add much context. Traditionally, users have wanted facilities which are different from those which the hardware provides. To solve this problem, a set of programs is usually written which translates the hardware capabilities into facilities which satisfy user requirements. This set of programs is referred to as an *operating system*. At this point, we will not attempt to further clarify or confuse the term. In Chapter 1 we present our view of what constitutes an operating system. We hope that by the end of the book the reader will be able to recognize an operating system when he sees one.

Most users do not need a thorough understanding of the operating system within which their programs run. They only have to abide by the constraints which the system imposes, such as memory partition size or job control language. Analogously, the Fortran programmer does not have to understand all the details of the Fortran compiler which he uses. There is general agreement, however, that an exposure to operating system principles is a necessary background for a "computer scientist." The purpose of this book is to provide that necessary background

to the reader. We do not attempt to define a general methodology for designing operating systems. Rather, we try to present in a concise and lucid manner the fundamental concepts and principles which govern the behavior of operating systems.

We have tried to keep our notation as consistent as possible with the literature. In the case of hardware this has been difficult to do, since manufacturers frequently use conflicting definitions. When in doubt, we have yielded to the force of large numbers and have adopted IBM's notation. We have not used any particular programming language to describe our algorithms, taking a good deal of liberty with control structures and basic operators where appropriate. Commonly used programming languages, such as PL/I, are cumbersome to use for descriptive purposes while more appropriate programming languages, such as PASCAL, are not widely known. Nuances in notation are clearly defined in the sections which introduce them.

This book is primarily intended as a text for a one-semester course directed toward fourth-year undergraduates and first-year graduate students, similar to the course I4 outlined by ACM Curriculum '68. Such a course has been taught at the University of Toronto with about a hundred students per year since 1969. The course (and the book) has been developed according to the recommendations of the COSINE task force on operating systems [Denning *et al.*, 1971]. We strongly recommend the report of the task force as a guideline to the lecturer who plans to institute a course on operating systems.

The students taking the course in our university have had a course on data structures and a course on language and compiler design. This background is not absolutely necessary, but it does give the students some maturity in the area of systems programming. We did not make any strong assumptions about the background of the students for whom the book is intended. We only presume some experience in computer science. Specifically, it is helpful if the students have written some long programs and have some idea of the problems encountered in interfacing with computer systems, such as fighting with job control language restrictions. It is also important that the reader have some knowledge of lists and queues. This material, which is outlined in Appendix I, can be presented in an early tutorial section for those students whose background is weak.

Many issues regarding the structure of operating systems are well understood. The problems of managing processes, processors, and memory, which are covered in Part I, have found extensive treatment in the literature for some time. Although there are still many open questions, these areas can be discussed in a reasonably neat way. Unfor-

tunately, not all the issues are quite so elegant. Part II deals with subject matter whose formal development is still in its incipient stages. Input–output, files, security, protection, design methods, reliability, performance evaluation, and implementation methods are all important aspects of operating systems. Despite the rather *ad hoc* flavor of many of the techniques associated with these topics, we feel they are nevertheless relevant and should be part of any introductory presentation of operating system principles.

Some of the basic concepts of operating systems come rather late in the book, such as virtual machines in Chapter 8. Our experience is that students find these concepts hard to appreciate before they have enough knowledge to relate them to examples in operating systems. Although some of the early material can be made more elegant using these concepts, we feel it may be too confusing in an introductory book.

There are many papers in the literature treating analytical problems of operating systems. Such work has generated important results which often give much insight into operating system behavior. We have tried to avoid most of these analytical results for two reasons. First, the results are more related to branches of applied mathematics than to the world of operating systems. Second, this material is more appropriate for a separate graduate course on modeling and analysis of operating systems than for an introductory course which emphasizes basic concepts. Theoretical work is well documented throughout the book; the annotated bibliography can serve as a guide to interested readers. We did, however, choose to make one exception. In Appendix II there is a short survey of computational structures, material relating directly to Chapter 2. Since most of the references for this subject are not widely available, we thought that it would be worthwhile to add a short discussion for the interested reader.

ACKNOWLEDGMENTS

Many of the ideas discussed in this book are the result of work done by the Project SUE group: J. W. Atwood, B. L. Clark, M. S. Grushcow, R. C. Holt, J. J. Horning, K. C. Sevcik, and D. C. Tsichritzis. Project members have influenced not only the material but also the spirit of the book. We very gratefully acknowledge their vast contributions and their continuous encouragement.

Many local students and faculty members have given us considerable help in editing early versions of the chapters. Their comments are responsible for many merits of the final manuscript. In particular, we would like to thank A. J. Ballard (Chapters 8 and 9 and Appendix III), C. C. Gotlieb (Chapter 7), R. C. Holt (Chapter 8), J. J. Horning (Chapters 1 and 2), R. N. S. Horspool (Chapter 4 and Problem Sets), E. Lazowska (Chapters 4 and 5), J. Metzger (Chapter 9), K. C. Sevcik (Chapters 3, 7, and 10), and F. Tompa (Chapters 4–6). The Annotated References are based on an earlier version by R. Bunt. We would also like to thank B. Liskov for her assistance in clearing up many foggy points in our description of the Venus Operating System in Chapter 10. Overall comments on a late version of the manuscript by B. W. Kernighan were quite helpful in locating and correcting a number of weak sections.

Special thanks go to our friend and associate J. R. Swenson whose careful reading of the final draft led to many changes on both a technical and pedagogical level. In particular, many of his ideas on memory management had a considerable influence on the organization and content of Chapters 4–6.

We are also indebted to many other colleagues, too numerous to mention by name, for exciting and informative discussions on various aspects of operating systems principles.

Finally, we would also like to thank M. Oldham, V. Shum, and P. Steele for clerical assistance rendered during various stages of the writing.

NOTES TO THE INSTRUCTOR

In parallel with the lectures, we have found that tutorial sections which cover a particular system in some detail are quite helpful in showing students how the ideas of the course fit together. In Chapter 10 we present two examples of operating systems. Further examples can be found in the books by Hoare and Perrot [1972] and Sayers [1971]. We recommend these discussions be held early in the term in order to motivate the material. Most students, especially undergraduates, have very little exposure to real operating systems. They can hardly be expected to be enthusiastic about solutions to problems which they do not appreciate.

The material in the book is complemented by problems at the end of the chapters, which serve many useful purposes. First, they elaborate on points which receive only cursory treatment in the text. Second, they help relate some abstract concepts to the real world of operating systems. Third, they point to papers in the literature for further reading. Many of the problems are necessarily open ended; that is, they have no single correct answer. Such problems are meant to provoke students to consider different alternatives. These problems can be used as topics for discussion sessions as well as for homework assignments.

The requirements of the course as given at the University of Toronto are as follows:

1. an examination in class to verify whether the students follow the course lectures;
2. a long take-home assignment with more substantial, thought-provoking problems which the students can develop in depth;
3. a project which brings the students closer to the real world of operating systems.

If the class is small, several short assignments may be substituted for the exam. Problems for both the test and assignment(s) can be taken from the chapter problems.

We will elaborate on the project requirement, because it presents a serious challenge to both the instructor and the students. It is not easy to design, supervise, and evaluate one hundred projects per year. We have tried several approaches.

1. An essay surveying some aspect of operating systems. In the problem sets we incorporate suggestions for essay topics. Essays from past years at the University of Toronto have been retained and put on reserve in the library for the benefit of future students. We have found essays to be particularly useful the first few years the course is given. They require a minimum of supervision, and they generate a nice local library on subjects which are too obscure to be included in the course lectures. In particular, developing a collection of descriptions of popular operating systems can be quite valuable. After the course is given several times, it is difficult to avoid duplicating topics. At this point, emphasis can be placed on other types of projects. However, essays should always remain as an alternative for some types of students, such as part-time students who are good programmers but do not have much exposure to the literature.

2. A joint effort in evaluating some aspects of the design or implementation of operating systems. The Rosetta Stone project, originally proposed by W. Wulf, for evaluation of system programming languages is an example of this type of activity [Wulf *et al.*, 1972]. A set of standard problems, such as process synchronization and memory management algorithms, are programmed by many students in two or three languages. In addition to the programs, students are asked for comments comparing the different languages. We have tried this type of project with limited success. The main problem is that it takes a good deal of maturity to evaluate languages. Since most students do not do much evaluating, the main value of the project for them is in learning a few new languages and in solving several small problems. We feel that this project, although

worthwhile, is probably better suited to a course in software engineering or systems programming than one in operating systems.

3. A "toy" operating system using a pedagogical, simulated hardware environment. In Appendix III we outline the basic issues in designing and supervising such a project. The students, working in teams of two or three, simulate a simple machine and then write a small operating system for it. We have tried this project repeatedly with great success. It can be somewhat expensive in computer time, but the students get some real experience in how complex systems are constructed. As a side benefit, the students get a taste of real programming problems by having to build a relatively large program in a team. They have to deal with project management, which has been the nemesis of more than one operating system design. At the University of Toronto we are currently using a toy operating system assignment which was designed by R. C. Holt for use with the TOPPS programming language [Holt and Kinread, 1972]. The TOPPS system [Czarnik *et al.*, 1973] can be run under any system which supports the XPL compiler writing system [McKeeman *et al.*, 1970]. Hopefully, a distribution tape and documentation will be available in the near future for universities which would like to try the project.

4. A programming project for a minicomputer in an environment of a software laboratory. This type of project can be very exciting, but it presumes the presence of a highly accessible minicomputer system. If the proper facilities are available, a number of interesting software packages would make good term projects, such as a simple executive, a spooling system, or a simple file system. There are several reports describing such an environment, for example Corbin *et al.* [1971] and Marshland and Tartar [1973]. In our university this approach will be adopted for an advanced course on operating systems. However, the supervision of a large number of projects of this type can be a problem.

5. A real contribution to the university computer center or to industry. We have a cooperative agreement with our computer center by which ten to fifteen students work on assignments proposed, supervised, evaluated, and ultimately used by computer center personnel. Both the center and the students are very enthusiastic, but we doubt that the number of participating students can be increased without serious organizational difficulties. To be successful, each student requires close supervision, and the number of available supervisors is limited. This project is particularly well suited to students who have little practical experience. Students with industrial experience generally do not choose to write one more systems program. For the same reason that a real programming

assignment is a thrill for an inexperienced student, a survey type of essay is very beneficial for an experienced systems programmer.

Availability of facilities will be the primary influence on which projects are most suitable for a given school. If there are any questions about the specifics of a given project, we will be happy to assist in any way we can.

CONTENTS

Appendix I. Data Structures

Appendix II. Computational Structures

Appendix III. A Toy Operating System

PART I

PRINCIPLES

The first five chapters of this book discuss the allocation of the basic resources of a computer system, namely, processors and memory. Chapter 1 provides a general overview of what operating systems do and how they are usually structured. Chapter 2 introduces the primary concept of *process* as a means for understanding the interactions among active computations in a computer system. Chapter 3 uses processes to show how a single processor can be allocated to several independent computations. Chapters 4 and 5 cover the allocation of main memory and ways of extending main memory using peripheral storage devices.

Methods of managing processes, processors, and memory have found extensive treatment in the literature. The fundamental issues are well understood. Therefore, the main problem is to present the material in a technically sound and pedagogically lucid manner.