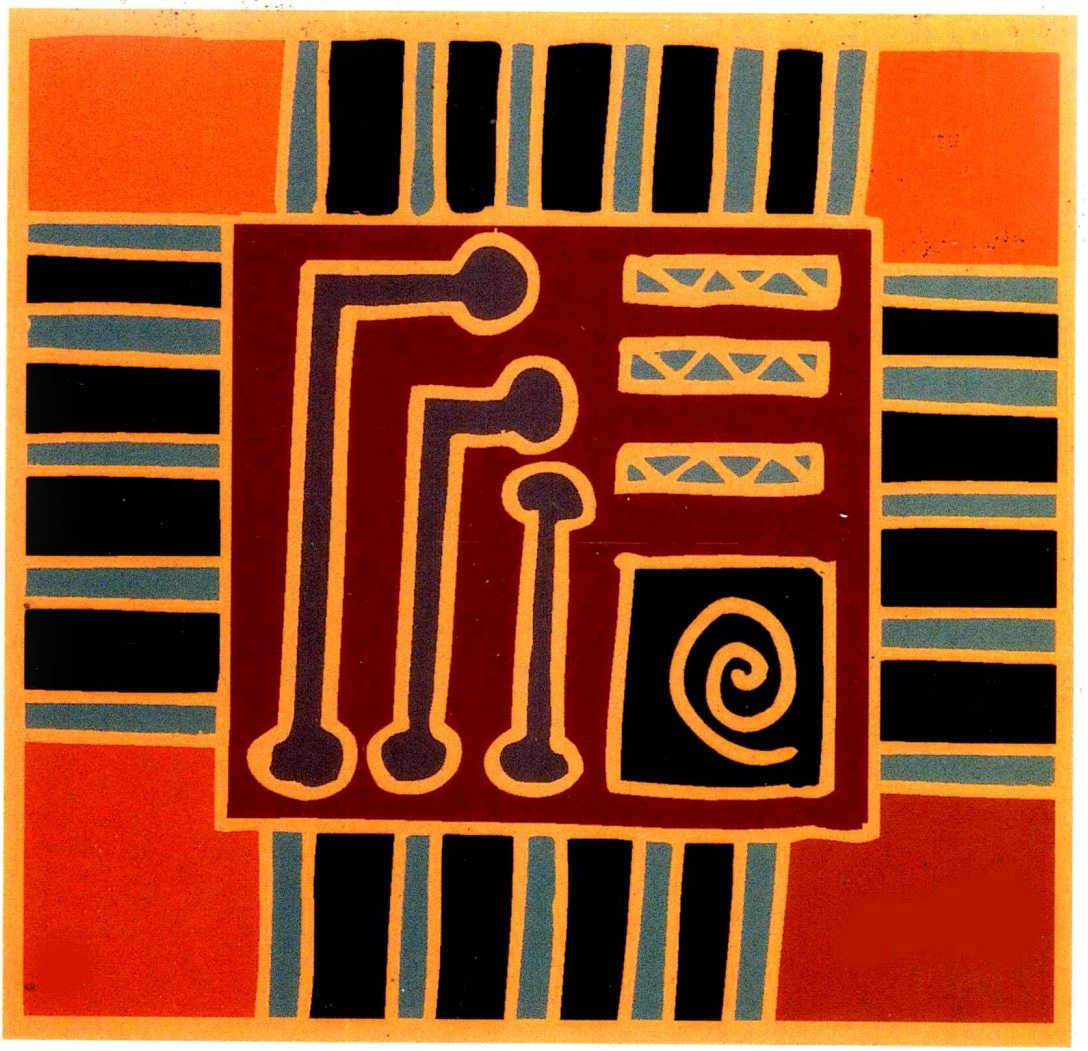

OPERATING SYSTEMS

A Design-Oriented Approach

Charles Crowley

OPERATING SYSTEMS

A Design-Oriented Approach

Charles Crowley
University of New Mexico

IRWIN
Chicago • Bogota • Boston • Buenos Aires • Caracas
London • Madrid • Mexico City • Sydney • Toronto

IRWIN **Concerned about Our Environment**

In recognition of the fact that our company is a large end-user of fragile yet replenishable resources, we at IRWIN can assure you that every effort is made to meet or exceed Environmental Protection Agency (EPA) recommendations and requirements for a "greener" workplace.

To preserve these natural assets, a number of environmental policies, both companywide and department-specific, have been implemented. From the use of 50% recycled paper in our textbooks to the printing of promotional materials with recycled stock and soy inks to our office paper recycling program, we are committed to reducing waste and replacing environmentally unsafe products with safer alternatives.

Irwin Book Team

Publisher: *Tom Casson*
Senior sponsoring editor: *Elizabeth A. Jones*
Senior developmental editor: *Kelley Butcher*
Project supervisor: *Lynne Basler*
Senior Production supervisor: *Laurie Sander*
Director, Prepress Purchasing: *Kimberly Meriwether David*
Compositor: *Interactive Composition Corporation*
Typeface: *10/12 Times Roman*
Printer: *Times Mirror Higher Education Group, Inc., Print Group*

Times Mirror
Higher Education Group

Library of Congress Cataloging-in-Publication Data

Crowley, Charles (Charles Patrick)
Operating systems : a design-oriented approach / Charles Crowley.
p. cm.
Includes bibliographical references and index.
ISBN 0-256-15151-2
1. Operating systems (Computers) I. Title.
QA76.76.063C77 1997
005.4'3—dc20 96–43708

Printed in the United States of America
2 3 4 5 6 7 8 9 0 WCB 3 2 1 0 9 8 7

PREFACE

In this book I have tried to approach the traditional, junior or senior level operating systems course in a new way. There are several areas where I have done things differently:

- *Describe the external operating system interface:* I start with a description of the system call interface to an operating system.
- *Use of code:* I have tried to steer a middle course between a concepts approach and a case study approach.
- *Development of concepts:* I have tried to show how the operating system concepts developed into their present form.
- *Design orientation:* I have tried to show how ideas from the design of operating systems relate to the design of other types of programs.

External Operating System Interface

Many students come to an operating system class without a clear understanding of what an operating system really does. These students should understand how operating system services are used before learning how these services are implemented. To deal with this, the book begins with a simplified, UNIX-like set of system calls. The book includes a discussion of these calls, example programs using these calls, and a simple shell program to integrate the examples.

Concepts or Case Studies

There have always been two approaches to the operating systems class. The first approach is the *concept* or *theory* approach which concentrates on the basic conceptual issues in the design of operating systems. These courses discuss each of the basic problems in operating systems design and the range of common solutions to those problems. The books are mostly text and diagrams with very little code. The second approach is the *case study* method which concentrates on an example operating system that is simple but complete. The books contain a lot of code and spend a lot of pages explaining the code in detail.

There are advantages and disadvantages to both approaches. Some people feel the ideal situation is to take both classes but this is rarely possible in an already crowded computer science curriculum so one is required to make a choice.

Use of code

I have tried to find a middle course between the two approaches. This book is basically a concepts oriented book with more code than is usual. I have found that seeing actual code allows the students to understand the concepts more deeply, feel more comfortable about

the material, and ask questions they wouldn't have thought to ask in a purely concepts oriented course. The code does not comprise a complete operating system however and it as simple as possible in order to reduce the amount of pages devoted to explaining it.

Development of Concepts

I have tried to show how these ideas developed. Many of the concepts in operating systems have developed over many years and the current solutions were developed slowly, in several stages. Each new solution had a problem that the next solution tried to fix. I think it helps the student to understand this development and see that these ideas were not brilliant flashes of insight that came out of nowhere but ideas that were improved by many people over many years. The development was a series of good ideas where each improvement made sense in the context in which it was developed. Seeing this development helps to understand why the solutions have their present form. In addition, it is useful to know the design constraints that caused solutions to develop into their present form because technological advances often change these constraints and old solutions that used to be inferior suddenly become practical again. Finally these developments give students examples of the design process through a series of potential but flawed solutions to a problem to a final solution that is acceptable.

Design orientation

Finally there is a concentration on design. In some ways designing an operating system is a pretty specialized activity having to do mainly with resource management. But many basic design ideas run through all designs and they show up in operating systems as much as anywhere else. Throughout the book I note places where we are presented with typical design problems. I abstract the operating systems related problems and solutions from the book into the general design problems and solutions and present them in a way that they can be applied to design problems in other areas of computer science.

Clearly it is not possible to cover all design topics and issues. I am striving for two things. First, I want to give the student an awareness of design issues, where they come up, which techniques to apply, how they can be generalized, etc. I do not present an organized survey of design techniques but a series of useful ones that come up in the context of operating systems. I hope to make the student aware of design and to enable the student to start doing their own generalizing about design. Second, I present a collection of useful design techniques that the students can use in their design toolkit.

Very few computer professionals will participate in the design of an operating system during the course of their careers. While it is important that students of computer science have a good foundation in the basic concepts in specialized areas such as operating systems, it is not necessary that every computer science student

understand all the details. However, there is a thread running through all areas of software engineering: the concept of design. There are many issued which are tackled during the design of an operating system which can be generalized and applied to other areas of computer science. In this book, I attempt to focus on these design issues and their implications for other areas.

I have oriented this book to provide a solid preparation for the larger design projects the student will encounter in later software engineering courses and as preparation for their career as a software professional designing, implementing and maintaining a wide variety of systems. This orientation also enables the operating systems course follows modern developments in the field of computer science. The interrelations between the separate areas of computer science are becoming more important. For example, in the area of high speed parallel machines, it is clear that it is necessary to think of the hardware, the operating system, and the programming language as a single system to get maximum performance. Optimization in any part of the system will have consequences for the other parts of the system.

The design techniques are noted in side bars as they come up and longer explanations of each design topic are placed in separate chapters from the operating system material. The instructor can structure a course with varying degrees of concentration on design aspects. The goal is that the design sections are independent of the main flow of the text and independent from each other. This will allow the instructor to pick and choose those design sections he or she finds to be useful.

USING THIS BOOK IN A COURSE

There is more material in this book than can be comfortably covered in a one-semester course. A number of the sections of the book have been marked with an asterisk. This indicates that they can be skipped with no loss of continuity in the presentation. In addition, all of the design chapters and design sidebars can be skipped with no loss of continuity. If you skip all the design chapters you can probably just cover the book in a semester. I expect that most instructors will choose to skip some of the design sections and some of the optional sections and teach a course that is about 90 percent operating systems and 10 percent design issues. Alternatively, the design issues could be covered in a separate one-unit course, strictly on design, that is taken along with the operating systems course.

ACKNOWLEDGMENTS

I want to thank my colleague Barney Maccabe who started out as my co-author on the book but had too many other commitments to work on the book. Our discussions lead to the conception of the book and the design orientation. I have talked with Barney on many aspects of operating systems and computer science and those discussions shaped many of my ideas.

I also want to thank John Brayer who used drafts of the book in several of his operating systems courses and put up with many typos, badly written sections, and unfinished sections. His comments helped in the development of the book.

I also want to thank my other colleagues in the Computer Science Department at the University of New Mexico. Continuing "in-the-hall" discussions with them have helped me formulate and improve my ideas.

Many students in CS 481 have used drafts of the books and have provided many useful comments and suggestions. I want to particularly thank Dave Rosenbaum who provided extensive comments on the entire draft, found numerous errors and typos and taught me some things about writing good English as well.

My treatment of the design chapters was greatly influenced by the design patterns developments in recent years and particularly the fine book by Gamma et el. (1995). I used their presentation format in the design chapters.

I want to thank all the people who reviewed this book while it was being developed. I corrected many problems and got many ideas for the presentation from their careful reviews and comments. The design chapters especially profited from their comments.

- Mustaque Ahamad, Georgia Institute of Technology
- Jim Alves-Foss, University of Idaho
- Anish Arora, The Ohio State University
- Brent Auernheimer, California State University - Fresno
- Anthony Q. Baxter, University of Kentucky
- Mahesh Dodani, University of Iowa
- H. George Friedman, Jr., University of Illinois - Urbana
- Tim Gottleber
- Stephen J. Hartley, Drexel University
- Giorgio P. Ingargiola, Temple University
- Stephen J. Krebsbach, South Dakota State University
- Donald S. Miller, Arizona State University
- Matt W. Mutka, Michigan State University
- Richard Newman-Wolfe, University of Florida
- Steve Reichenbach, University of Nebraska - Lincoln
- Bernhard Weinberg, Michigan State University

Finally my wife, Ella Sitkin, put up with many bad moods and excuses that I was "too busy" during the years this book was developed. As if that wasn't enough, she also edited several chapters.

CONTENTS