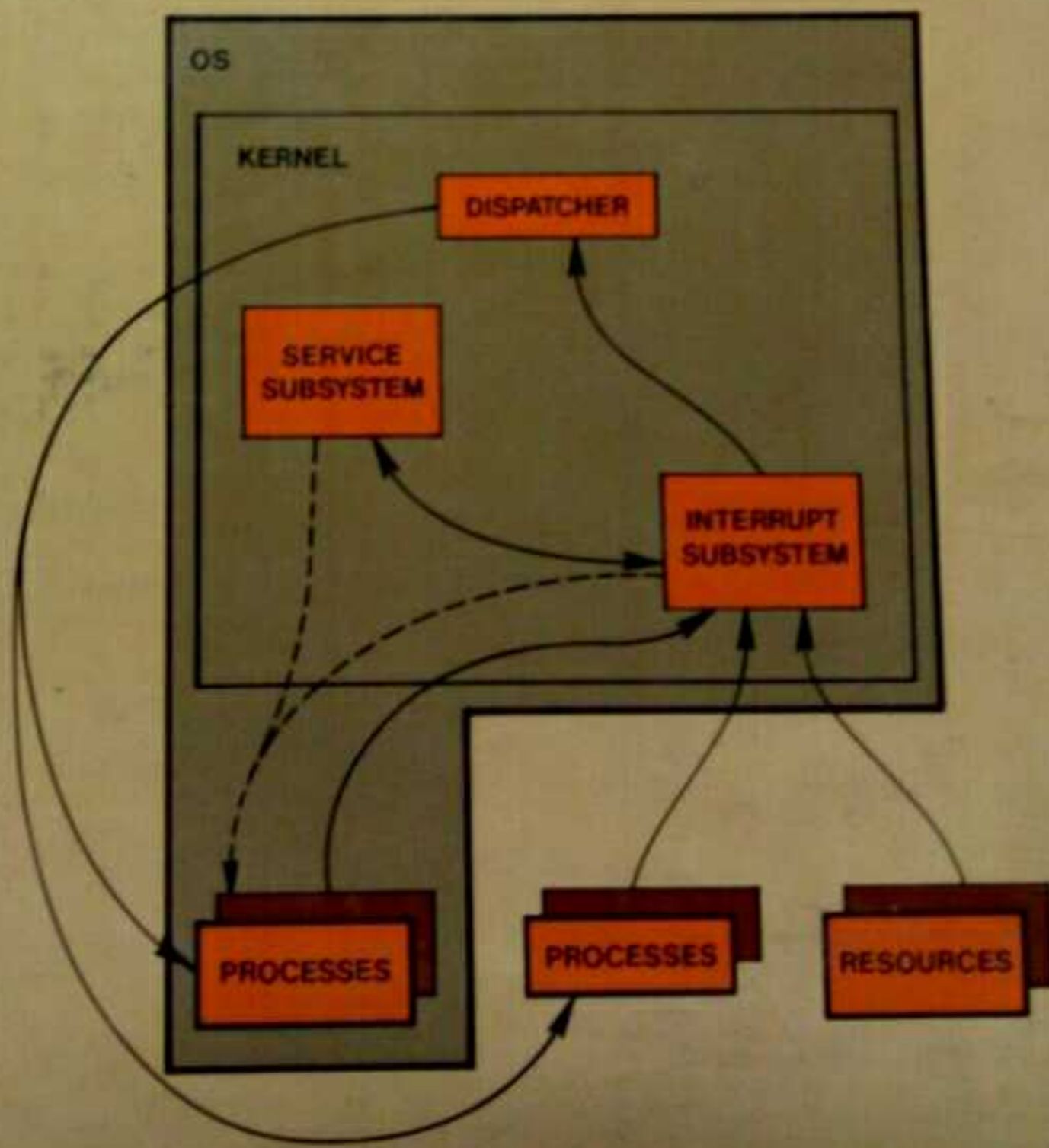


Raymond W. Turner

Operating Systems

Design and Implementations



183

501-642

TUR

Operating Systems

Design and Implementation

Raymond W. Turner

*Senior Scientist, GHD Corporation
Houston, Texas*

507
SEMINAR LIBRARY
Department of Computer Science
UNIVERSITY OF KARACHI

1-6-95

Macmillan Publishing Company
New York

Collier Macmillan Publishers
London

Contents



1	Introduction	1
1.1	Operating Systems / 2	
1.2	MOS Overview / 4	
1.3	MOS Data Structures / 7	
1.4	Host Hardware Environment / 9	
1.5	An Alternative Software Structure / 11	
	Problems / 12	
2	Multiprogramming and Basic Memory Management	15
2.1	Process Management and Basic Data Structures / 18	
2.2	Dynamic Memory Management Strategies / 20	
2.3	Dynamic Memory Management Algorithms / 27	
2.4	Boundary-Tag Method / 30	
2.5	Comparison of Algorithm Performance / 37	
2.6	Concurrency in Memory Management / 41	
	Problems / 42	
3	Multiprogramming and Virtual Memory	45
3.1	Program Overlaying / 46	
3.2	Memory Partitioning, Program Relocation, and Memory Protection / 49	
3.3	Process Swapping in MOS / 51	
3.4	Program Segmentation / 55	
3.5	Paged Memory Systems / 58	
	Problems / 63	
4	Basic Input/Output Programming	67
4.1	Direct Input and Output / 67	

5 Interrupts and Input/Output

95

- 5.1 Interrupts / 95
 - 5.2 Process- and Device-State Changes / 97
 - 5.3 Data Structures and Processing for Input and Output / 99
 - 5.4 Program Logic for Multiprogrammed Input/Output / 107
 - 5.5 A Final Look at Buffering / 116
 - 5.6 Alternative Input/Output Queuing Strategies / 120
- Problems / 122

6 Cooperating and Communicating Concurrent Processes

129

- 6.1 Processes, Subprocesses, and Process Relationships / 129
 - 6.2 Messages / 133
 - 6.3 Process Creation / 140
 - 6.4 Process Traps / 143
 - 6.5 Implementing System Functions as Processes / 152
- Problems / 154

7 Competing Concurrent Processes

157

- 7.1 Critical Sections and Race Conditions / 157
 - 7.2 Lock and Semaphore Operations and Options / 161
 - 7.3 LOCK and UNLOCK Algorithms / 167
 - 7.4 Deadlocks / 171
 - 7.5 A Final Look at Locks / 175
- Problems / 176

8 Timer Management

183

- 8.1 The Interval Timer / 183
 - 8.2 Time-Related Services / 186
 - 8.3 Timer-Interrupt Processing / 192
 - 8.4 Using Time Management Functions / 196
 - 8.5 Real-Time Processing / 198
- Problems / 199

9 Process Scheduling and the Dispatcher 203

- 9.1 System List Management / 203
- 9.2 Communication with the Dispatcher and Event Handling / 207
- 9.3 Scheduling Policies / 213
- 9.4 Implementing Process Scheduling in MOS / 215
- 9.5 A Final Look at Events and Internal Operating System Communications / 218
Problems / 219

Appendix

A Process Design Language Definition 225

- A.1 Structure and Notation / 226
- A.2 Implementation / 226
- A.3 Nonrepeating Structures / 227
- A.4 Repeating Structures / 231
- A.5 Module Abstraction / 233
- A.6 Example / 234

B MOS Simulator 235

- B.1 Overview / 235
- B.2 Simulated Hardware and Data Structures / 236
- B.3 Simulated Machine Instructions / 240
- B.4 System Calls and Interrupts / 241
- B.5 MOSSIM Subroutine Interface Summary / 242

Bibliography and Reading List 247

Index 253

CHAPTER 1

Introduction

This is a book about operating systems and systems programming. What is systems programming? Some claim that it is programming done by "systems programmers" working for a computer manufacturer developing software delivered with the computers. If this definition is accepted, not only is operating system development encompassed by systems programming, but also the development of text editors, BASIC interpreters, COBOL compilers, linkage editors, scientific subroutine libraries, word processors, data base management systems, and so on. As a matter of fact, since most manufacturers have many standard industrial and commercial software applications packages available to enhance the marketability of their products, development of such programs as general ledger, inventory control, and energy management might also be considered systems programming.

Obviously, this definition is too broad to define topics to be covered adequately in a single-semester course. How about defining systems programming as the development of "nonapplications" programs? But what is an applications program, as opposed to a systems program? It is a program that executes under the control of the operating system, generally in a nonprivileged mode. But under this definition, all programs listed in the preceding paragraph, except the very heart of the operating system itself, could be considered applications. Should text editors and compilers be classified as applications programs? Systems programming might, therefore, relate to the programming of the operating system itself and nothing more. (Thus it follows that the only "systems programmers" are the programmers who develop or maintain operating systems/

Now we have a definition that is probably too restrictive. For the purposes of this text, **systems programming** will be defined as programming that uses techniques similar to those commonly found in operating systems, even though the use may not be in direct support of the operating system. Generally, these techniques will be used to develop software that can provide a programming