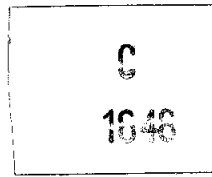


BIBLIOTHEQUE DU CERIST



FILE DESIGN & PROGRAMMING

W. WESLEY PETERSON
ART LEW



FILE DESIGN AND PROGRAMMING

W. Wesley Peterson and Art Lew

University of Hawaii at Manoa

John Wiley & Sons, New York • Chichester • Brisbane • Toronto • Singapore

Preface

Files are second in importance in computer science to algorithms and techniques for their implementation as programs. In all systems that handle large amounts of information, files play an essential role. A computer without a file system is hardly more important than a large calculator. Thus, it is essential for computer scientists and systems analysts and designers to have a good fundamental understanding of files. In this book we provide this basic information on files, including methods for their analysis and design, and algorithms and programming language facilities for their processing.

At the University of Hawaii, we have offered a required one-semester course on files for the past eight years. This book is based on course notes that evolved over these years. The course is roughly equivalent to the course CS 5 *Information to File Processing* in the ACM Curriculum 78, and the book is most suitable for that course. At the University of Hawaii, we cover the entire book, except for Appendices F and G and some of the finer details, in one semester. Our students have at least sequential I-O programming experience with Pascal, COBOL, and FORTRAN, but very limited knowledge of data structure analysis and design when they take the course. Students with a good introduction to data structures but no experience with PL/I, COBOL, or FORTRAN should be about as well off. We find that a writing knowledge of one language provides a sufficient reading knowledge of others, especially when the programs perform the same tasks, to warrant the presentation of file algorithms in several different languages. This kind of presentation also makes this book ac-

cessible to those having a variety of backgrounds. Those interested in only a single language will find that the additional information reinforces and clarifies the most important concepts, especially the relationships between languages and operating systems.

We have restricted ourselves to programming languages that are defined by national standards and that include random-access file handling techniques. These languages are representative of most others, and there seems to be no need to include material that is dependent on a particular language implementation. Ada, C, and Pascal are clearly very important languages, but they do not have file I-O capabilities included in standardized definitions that are comprehensive enough to justify including them. In Pascal, for example, file I-O is not specified by programming language statements but, instead, by calls to implementation-dependent procedures.

In this book, we usually describe the *logic* of algorithms in Pascal form, but use COBOL, PL/I, and FORTRAN (in Chapter 8 and the Appendices) to provide actual programming examples that can be used by the reader with little if any modification in whatever computer system is available. We believe it is essential that the reader try running one or more of these actual programs and experiment by perturbing them in various ways. It also would be instructive for readers to translate these programs to non-standard implementations and various other languages. Where this book is used as a textbook, these would make good programming assignments.

We also place considerable emphasis in this book on calculating timing and space requirements for files. These are not necessary for writing file processing programs, and for many applications the time and space requirements are obviously small and need not be calculated before a system is developed. This is certainly true of students' programming exercises. However, it is clearly important for a computer scientist or systems analyst to be able to predict the performance of a large system and to do a good deal of analysis and design before any appreciable investment is made in producing it. Therefore, a large portion of this book consists of quantitative examples illustrating the kinds of calculations that can be made with a minimum amount of mathematics. (No calculus is needed.)

Part 1 of this book is devoted to file analysis and design techniques. Part 2 and much of the Appendices are devoted to file processing algorithms and programming language facilities to implement these algorithms. The two parts of the book are fairly independent and can be covered in either order with a minimum amount of cross-referencing. This might be appropriate for two single-quarter courses, or for independent study. However, to teach both parts in a one-semester course requires covering the material concurrently so that students can start working on simple programming exercises while studying the analysis

and design concepts, and then gradually increase the complexities of their programs while learning more and more of the principles.

We gratefully acknowledge the support of the Department of Information and Computer Sciences at the University of Hawaii, without which this book would not have been possible. Our production of the first draft of this book was supported in part by the National Bureau of Standards. The nice unified treatment of file maintenance in Chapter 7 is based on an unpublished report by Diana Foley. Anne Niethammer contributed a number of suggestions. We received a good deal of support from our secretary Ethel Shintaku and her student helpers. Finally, we have both learned a great deal from our students. We owe sincere thanks to all of these people.

W. Wesley Peterson
Art Lew

Contents

PART 1 FILE ANALYSIS AND DESIGN 1

1

Introduction 3

- 1.1 Files in Computer Systems 3
- 1.2 Information Storage Devices in the Computer System 5
- 1.3 Basic File Concepts 6
- 1.4 Overview of this Book 8

2

Magnetic Tape 11

- 2.1 Introduction 11
- 2.2 Tape Capacity 12
- 2.3 Blocking 14
- 2.4 Tape Timing 17
- 2.5 Other Tape Concepts 19

3

Rotating Storage 24

- 3.1 Introduction 24
- 3.2 Disk Capacity 29
- 3.3 Head-per-Track Disk and Drum Timing 33
- 3.4 Moveable Head Devices 38

- 3.5 Seek Time for Successive Accesses in Same File 40
- 3.6 Seek Time for Access from One File to Another 43
- 3.7 Complete Access Time Calculations 48

4

Processor-Storage Interface and Buffering 54

- 4.1 Introduction 54
- 4.2 Typical Computer Organization 55
- 4.3 Direct I-O 55
- 4.4 I-O Processors 58
- 4.5 IBM 370 Channel Programming 60
- 4.6 Simple Input Buffering 62
- 4.7 Simple Output Buffering and Multiple Buffering 67
- 4.8 Blocking and Buffering with Tapes 72
- 4.9 Blocking and Buffering with Disks 76
- 4.10 Other Topics 83

5

File Structures 88

- 5.1 Data Structures 88
- 5.2 Arrays, Relations, and Tables 89
- 5.3 Linked Data Structures 92
- 5.4 Linked Lists 93
- 5.5 Trees 95
- 5.6 Tree Searching and Sorting 98
- 5.7 B-Trees 100
- 5.8 Multilists and Rings 107
- 5.9 Inverted Files and Cellular Multilists 110
- 5.10 Database System Data Structures 111
- 5.11 Hierarchical Database Structure 114
- 5.12 Network Database Structure 117
- 5.13 Relational Database Structures 124

6

File Design 131

- 6.1 Files 131
- 6.2 Common File Accessing Requirements 132
- 6.3 Sequential Files 134
- 6.4 Random-Access Files 134

6.5	Indexed Sequential Files	135
6.6	Searching Sequential and Indexed Sequential Files	138
6.7	Hashed Files	140
6.8	Comparison of Indexed Sequential and Hashed Files	150
6.9	Designing a Random-Access File	151
6.10	Record Structures	156
6.11	File Compression	160
6.12	File System Allocation	161

PART 2 FILE PROCESSING AND PROGRAMMING 171

7

File Processes: Sorting and File Maintenance 173

7.1	Introduction	173
7.2	File Sorting	173
7.3	File Maintenance	183
7.4	Edit Checking	186
7.5	Batch Update Processing	189
7.6	On-Line Update Processing	197
7.7	Report Writing	201

8

File Programming 210

8.1	Introduction	210
8.2	File Creation and Deletion	211
8.3	File Declarations and Open Statements	212
8.4	Stream File Processing	214
8.5	Sequential Processing of Records	215
8.6	Random Processing of Records	216
8.7	Error Conditions	217
8.8	Blocking and Buffering	217
8.9	I-O Statements in COBOL	218
8.10	Procedure Division Statements for Sequential Files	221
8.11	Relative Files in COBOL	224
8.12	Indexed Sequential Files in COBOL	228
8.13	File Statements in PL/I	231
8.14	Stream and Sequential Files in PL/I	233
8.15	KEYED Files in PL/I	235
8.16	File Programming in FORTRAN	238

- 8.17 Sequential File Programming in FORTRAN 239
- 8.18 Direct Access Files in FORTRAN 242
- 8.19 Programming of Hashed Files 243

APPENDICES 248

A

Data Coding 248

- A.1 Introduction 248
- A.2 Character Data 248
- A.3 Binary Data 251
- A.4 Packed Decimal Code 252
- A.5 Floating Point Representation 252
- A.6 Conclusion 254

B

COBOL Program Examples 256

- B.1 Batch Update Program 256
- B.2 Main Program for On-Line Update 261
- B.3 Relative File I-O Procedures 265
- B.4 Indexed Sequential File I-O Procedures 268
- B.5 Hashed File I-O Procedures 271

C

PL/I Program Examples 281

- C.1 Batch Update Program 281
- C.2 Main Program for On-Line Update 286
- C.3 Relative File I-O Procedures 291
- C.4 Indexed Sequential File I-O Procedures 295
- C.5 Hashed File I-O Procedures 299

D

FORTRAN Program Examples 309

- D.1 Batch Update Program 309
- D.2 Main Program for On-Line Update 316
- D.3 Relative File I-O Procedures 323
- D.4 Hashed File I-O Procedures 326

E

Command and Job Control Languages 334

- E.1 Introduction 334
- E.2 Executing Programs 336

- E.3 **File Handling** 337
- E.4 **Command Procedures** 341

F **Cache Memory and Virtual Storage** 345

- F.1 **Introduction** 345
- F.2 **Cache Memory** 346
- F.3 **Virtual Storage** 346

G **An Example of an I-O Processor** 351

- G.1 **Introduction** 351
- G.2 **8085 Machine Language** 351
- G.3 **Direct I-O** 355
- G.4 **I-O Processors** 358

- References** 362
- Index** 369