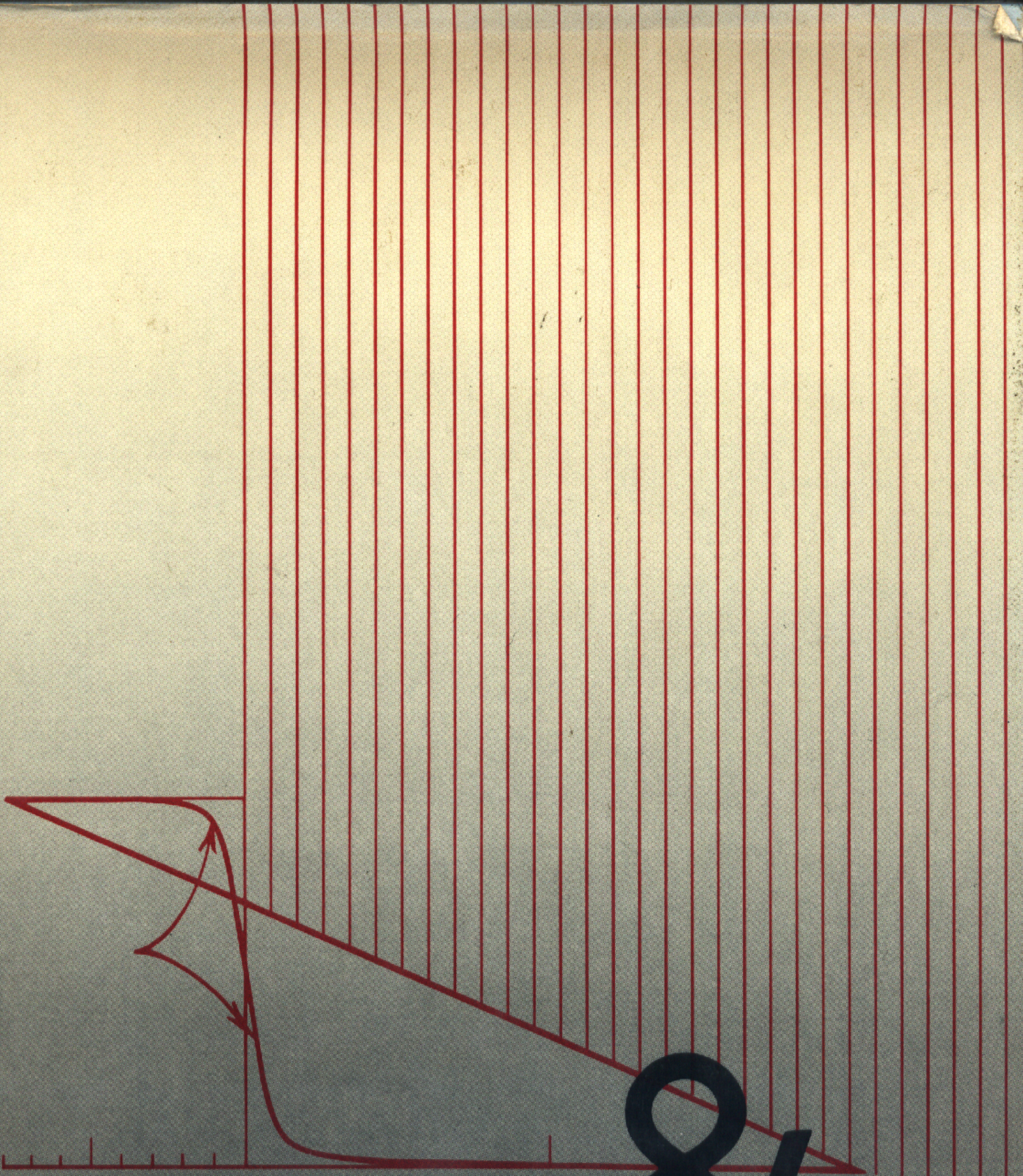

FILE DESIGN & PROGRAMMING

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 accessible 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 nonstandard 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