FILE STRUCTURES FOR ON-LINE SYSTEMS

DAVID LEFKOVITZ, Ph.D

Associate Professor
Moore School of Engineering
University of Pennsylvania

Staff Consultant
Computer Command and Control Company
Philadelphia

Spartan Books

Macmillan Education

Dedicated to the memory of my father
Joseph Lefkovitz

ISBN 978-1-349-00695-3 *ISBN 978-1-349-00693-9 (eBook)*
DOI 10.1007/978-1-349-00693-9
Library of Congress Catalog Card Number 68-26073

Hayden Book Company, Inc.
50 Essex Street, Rochelle Park, New Jersey 07662

Softcover reprint of the hardcover 1st edition 1969 978-0-333-10113-1

Spartan Books are distributed throughout the world by Hayden Book Company, Inc., and its agents.

3 4 5 6 7 8 9 PRINTING

75 76 77 78 YEAR

PREFACE

In June, 1967, The Professional Development Seminars of the Association for Computing Machinery offered a one-day course on File Structures for On-Line Systems to its membership, and to the data processing community at large. These seminars were given twenty-four times throughout the United States during 1967 and 1968, and it was the continued and widespread interest in this subject that influenced the author to produce this book. The subject is approached here in much the same way as presented in the lectures, although a number of topics are treated more comprehensively than was possible in the lectures, a more complete treatment of memory requirement and response time formulations is presented, and the initial chapters provide a perspective from which one can relate over-all system functional requirements to file structure design criteria.

The structuring of files for on-line systems is pertinent to two broad categories of use: on-line programming systems and on-line information storage and retrieval systems. The demands of the latter are usually more stringent because of the significantly greater number of keyed entries to the file, the requirement for multiple key access (i.e., logical key combinations) to the file, and a considerably higher volume of data. The response time requirements of the programming systems may be more critical than those of information systems, but the simplicity of access modes tends to offset file structuring problems caused by this requirement. An example of an on-line programming system file structure problem is the construction of an output file.

Since the quantity of output generated by a given program in some time interval cannot be known in advance (except possibly within a system or programmer imposed upper limit), storage allocation on a direct access storage device (DASD) for the output data can most expeditiously be made via a chained list structure. A file with a name (or key) that can be related to a program and a terminal device is opened on the DASD with a relatively small quantum of allocated space, such as a seventy-two character line of type or some small number of lines. Should the program generated data exceed this buffer capacity, another such buffer could be linked by a chained address, since the contiguous space on the disk may have been assigned, in the interim, to another program. This list structure is what was first described in the literature as a threaded list.[8] It is easy to construct, conserving of memory space, and fast to retrieve because *every* record on the list (i.e., every link of the chain) is a desired record for subsequent processing; however, as soon as multiple key entry to a file is allowed, the situation becomes somewhat more complex, because every record on the list is not necessarily required, and the problem of list retrieval efficiency must be faced. To extend this same example, if it is assumed that an additional key denoting output device were attached to each output buffer (in which case these records assume a logical significance since the entire file of records is no longer a continuous data stream), and if the situation were to arise wherein it is desired to access "the next output record for Program X to be printed on the line printer," then a number of alternatives are possible for the organization of this file, including, (1) threading the records on a Program tag and qualifying it after accession by a Terminal tag, (2) threading the records on a Terminal tag and qualifying by a Program tag, or (3) threading the records on both tags and accessing records from the shorter list. The latter file structure is often referred to as a multiple threaded list or Multilist.[10] This example illustrates the difference between the single and multiple key record, but it is more typical of generalized information storage and retrieval system file structuring requirements than of on-line programming requirements. The generation and maintenance of input, intermediate, output, and program files for on-line programming systems generally involves at most the simple threaded list and can frequently make use of a serial structure. The file structure of information systems, on the other hand, is often multiple key, which imposes design considerations related to speed of initial, successive, and total system response, speed of update, quality of

presearch statistics, ease of programming, and list structure overhead in time and space.

This book, therefore, uses the information system as a frame of reference for discussion of on-line file structures, largely because file management techniques for programming systems can easily be drawn from those developed for the former. The first chapter of the book relates file structures functionally to the requirements of automated information systems. In the second chapter direct access storages are divided into three classes, and concepts relating pertinent hardware features of these devices to software design are introduced. The third chapter distinguishes the concepts of information structure and file structure, where the former is defined to be an inherent property of information as it may be generated and organized by the people who utilize the system, while the latter is the means by which the automation designer organizes these data into various files for storage in the random access memories and retrieval and update by the computer programs. Two basic types of information structure, *associative* and *hierarchic,* are analyzed in terms of alternative file structures. This chapter views the file design process from inside out; that is, the construction of record and subrecord controls for the file structure are examined, which satisfy the information structure requirements. In Chapter IV an outside in view of the file structure is presented by means of a prototype query language that might appear as the interface between the users and the file system.

In Chapter V, all of the techniques to be described in the remainder of the book are classified, and their general characteristics and principal design applications are discussed. Also presented in this chapter is the notion of the two-stage retrieval process: Directory decoding and file search. In Chapter VI, various techniques for the design and construction of Directories and their decoders are presented along with storage requirement and timing formulations; in Chapter VII, methods of structuring or organizing the files containing the searchable records are presented along with storage and timing formulations. The final chapter of the book deals with the on-line updating and maintenance of the various file organizations discussed in Chapter VII.

The primary objective of the book is to present the concepts of file design as well as sufficiently detailed descriptions of techniques that have been used to implement these designs. In presenting concepts, the areas of maneuver or design tradeoff are emphasized, particularly with regard to system response, cost, and programming complexity. The description of techniques is at a level that a somewhat experienced

programmer should have little difficulty in implementing, and, in addition, the basic elements of design enable either the programmer or the analyst to modify any of the specific techniques to suit a given problem.

The author is indebted to a number of persons who have participated in various ways to the accumulation and evaluation of material contained in this book. Firstly, the author acknowledges the contributions of Dr. Noah S. Prywes who supervised much of the government sponsored research and development work under which many of these file structuring techniques were developed and tested. The author's colecturer in the ACM seminars, Thomas Angell of Computer Command and Control Company, has contributed through discussion, evaluation, and improvement of presentation of the material. James Adams, ACM Professional Development Seminar coordinator was primarily instrumental in the organization and promotion of the lectures and, in addition, contributed substantially to their improvement through evaluation of the attendee critique sheets. Others who have made contributions through discussion, comment, and programming design and application are James Russell of Computer Command and Control Company, and Ruth Powers of the University of Pennsylvania.

The author also wishes to thank Mrs. Esther Cramer and Mrs. Lois Porten for their excellent typing of the several drafts, and Miss Mary Jane Potter for drafting all of the book's illustrations.

CONTENTS