DOCUMENT RESUME

ED 078 880 LI 004 415

AUTHOR Hsiao, David K.
TITLE Logical Access Control Mechanisms in Computer Systems.
INSTITUTION Ohio State Univ., Columbus. Computer and Information Science Research Center.
SPONS AGENCY Office of Naval Research, Washington, D.C.
REPORT NO OSU-CISRC-TR-73-4
PUB DATE Jul 73
NOTE 27p.;(11 References)

EDRS PRICE MF-$0.65 HC-$3.29
DESCRIPTORS Computer Programs; *Computers; *Computer Storage Devices; *Confidentiality; *Electronic Data Processing; On Line Systems; Security; Time Sharing

ABSTRACT

The subject of access control mechanisms in computer systems is concerned with effective means to protect the anonymity of private information on the one hand, and to regulate the access to shareable information on the other hand. Effective means for access control may be considered on three levels: memory, process and logical. This report is a review of the logical access control mechanisms implemented in two computer systems. The first system, PSF, is an on-line, multi-access operating system with an advanced data base management capability on an IBM 7040/PDP/8/DEC 338 coupled computer configuration. The second computer system, EDME, is a time-sharing operating system with extended data base management capability on a RCA Spectra 70/46 virtual memory computer. (A related document is LI004414.) (Author/SJ)

(OSU-CISRC-TR-73-4)

LOGICAL ACCESS CONTROL MECHANISMS IN COMPUTER SYSTEMS

by

David K. Hsiao

Work performed under

Contract N00014-72-C-0391, Office of Naval Research

Computer and Information Science Research Center

The Ohio State University

Columbus, Ohio 43210

July 1973

PREFACE

This work reported herein was, in part, supported by the Office of Naval Research under Contract N00014-72-C-0391.

The paper was delivered as an invited talk in the ADP Secure Data Sharing Conference sponsored by ONR/NSRDC, Washington, D.C., September 26-28, 1972. It was included in the Proceedings of the conference July 1973.

It is published by The Computer and Information Science Research Center of The Ohio State University which is an interdisciplinary research organization consisting of the staff, graduate students, and faculty of many University departments and laboratories.

Table of Contents

I. INTRODUCTION

The subject of access control mechanisms in computer systems is concerned with effective means to protect the anonymity of private information on the one hand, and to regulate the access to shareable information on the other hand. Effective means for access control may be considered on three levels: memory level, process level and logical level.

At the memory level, access control mechanisms are those which regulate access to memory in terms of units of memory. The main point is that protection is of the containers, i.e., the memory units, not the contents. As a consequence, everything inside the container is subject to the same access control as the enclosure itself. Furthermore, the contents are safe only as long as they are kept in the same containers.

Typically, physical memory protection schemes employ memory bounds registers or storage protection "keys" which control access to bounded memory areas. Other, more sophisticated, schemes are possible. The idea of having an $m \times m$ matrix of control bits to keep track of access rights to m memory areas has been advanced [1]. For example, an entry A_{ij} would determine the access rights to i-th area from the j-th area. The A_{ij} may correspond to various access rights such as read-only, read/write, execute-only and privileged mode.

In general, one user's access rights to an area may differ considerably from another user's access rights to the same area. In a multi-programming and shared data base environment, the system must therefore provide dynamically different access matrices for different users. The use of virtual memory may, therefore, enhance the implementation of the matrix scheme. Here, page and segment tables are consulted by the hardware at instruction decoding time. Each user is assigned his own tables and therefore cannot get into a segment of memory which does not have an entry in those tables. As a result, sophisticated schemes such as the access matrix are more easily implemented with virtual memory. A good

case of such implementation is presented in the Conference by our first speaker [2]. Yet, even in virtual memory, we note that the protected areas are again units of memory.

The second level of access control is called process access control. A process is simply a set of programs and their associated data. Thus, unlike memory protection, the notion of process protection and control is concerned with access to and protection of programs. To this end, we must develop mechanisms which can determine when and under what conditions programs can pass control from one to another. In other words, the mechanisms must be able to monitor the execution of programs in terms of their calls, returns and transfer of parameters.

An elaborate process access control mechanism was proposed and known as the "ring mechanism" [3]. This concentric ring mechanism allows one program to give control to another without violating any of the access control rights of either program, thereby safeguarding each program's working tables, data, intermediate results, etc. Conceptually, the concentric ring mechanism requires the user to arrange his processes hierarchically, i.e., processes at the lower part of the hierarchy (i.e., outer rings) have less privileged access rights. This mechanism can be implemented in a computer whether or not it has virtual memory. Therefore, one should not indulge in the misconception that virtual memory protection and process protection are one and the same.

The highest level of access control is logical. We feel that, in a large data base environment, the user will first organize his data into some logical structure. He will then refer to his structured data in terms of logical entities such as fields, arrays, records and files. The important point is that these entities are logical units of information which may have little resemblance to their physical or virtual storage images. By allowing the user to associate access control requirements and protection measures with the logical units, the access control mechanism can facilitate direct control and protection of the information regardless of the whereabouts of that information. Furthermore, the

mechanism does not require the user to be familiar with the physical or virtual storage structure of the computer system.

Logical access control mechanism must therefore have the facility for the user to describe his shareable and private data in terms of logical entities of the data base, to assign access rights and protection requirements to these entities, to determine the collections of these entities and the types of access that other users may have, and to incorporate additional authentication and checking measures in terms of procedures.

This paper is basically a review of the logical access control mechanisms implemented in two computer systems. The first computer system, known as the PSF, is an on-line, multi-access operating system with an advanced data base management capability on an IBM 7040/PDP/ 8/DEC 338 coupled computer configuration (see Fig. 1 and Fig. 2). Research and development period of PSF covers the years between 1965 and 1968. It was in operation until 1971. Documentation on PSF's data base management and access control capability can be found in [4]. For a brief introduction to PSF's access control mechanism the reader may refer to [5]. Subsequent evaluation and generalization of the PSF logical access control mechanism can be found in [6,11]. The second computer system, known as the EDMF, is a time-sharing operating system with extended data base management capability on a RCA SPECTRA 70/46 virtual memory computer (see Fig. 3 and Fig. 4). Research and development period of the EDMF covers the years from 1968 to 1971. It is presently operational. Documentation on EDMF's access control mechanism can be found in [7]. For a brief introduction to EDMF, the reader may refer to [8].

The software experimentation on logical access control mechanisms in these computer systems is aimed to understand the working principles and requirements of the mechanisms with a view toward their hardware implementations. It is hoped that a good understanding of the experimentation can lead to a better and useful implementation of logical access control mechanisms in hardware. To this end, we have chartered our present research efforts.

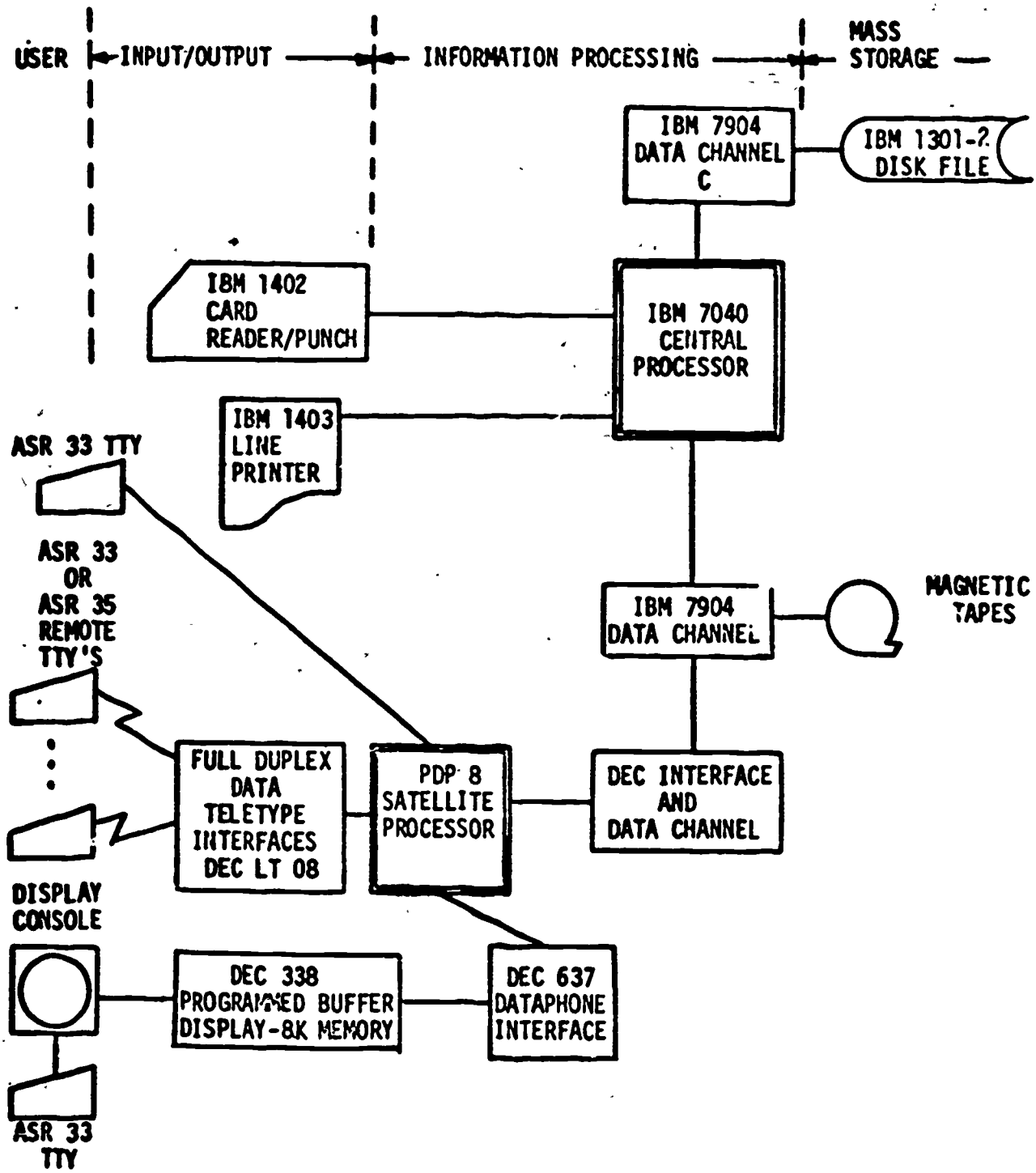


FIG. 1 PSF - EQUIPMENT CONFIGURATION

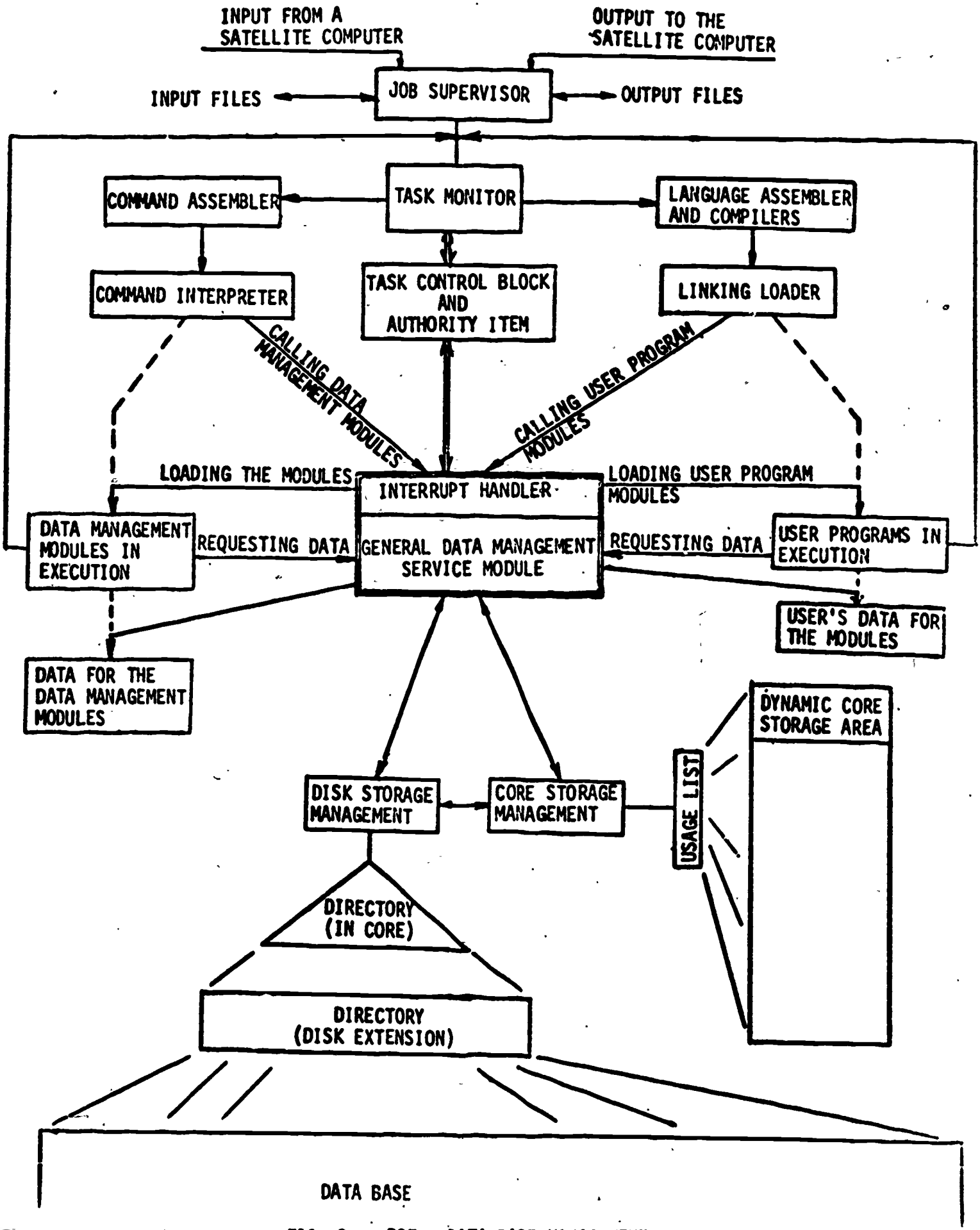


FIG. 2 PSF - DATA BASE MANAGEMENT SYSTEM ARCHITECTURE

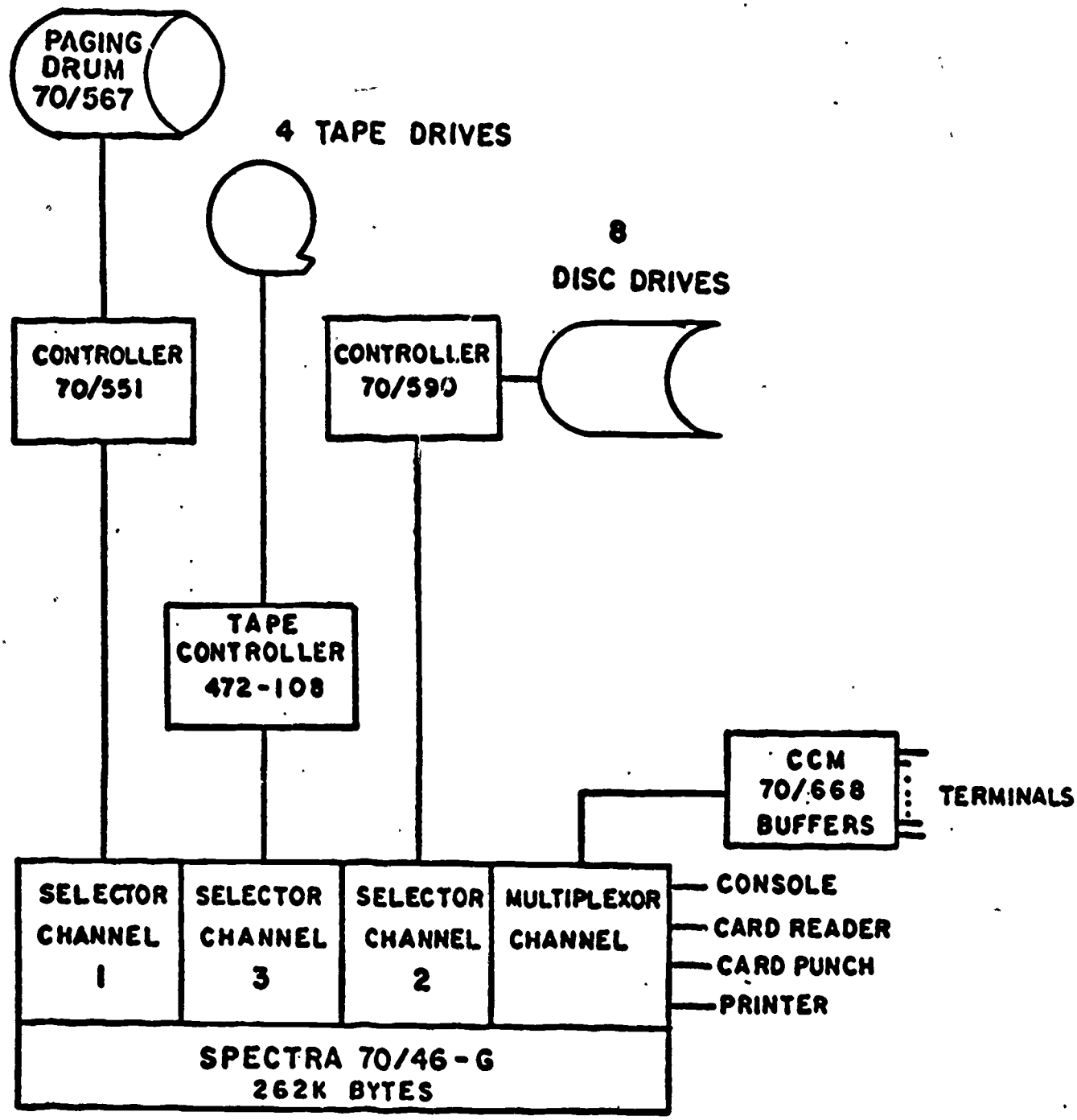


FIG. 3 EDMF - EQUIPMENT CONFIGURATION

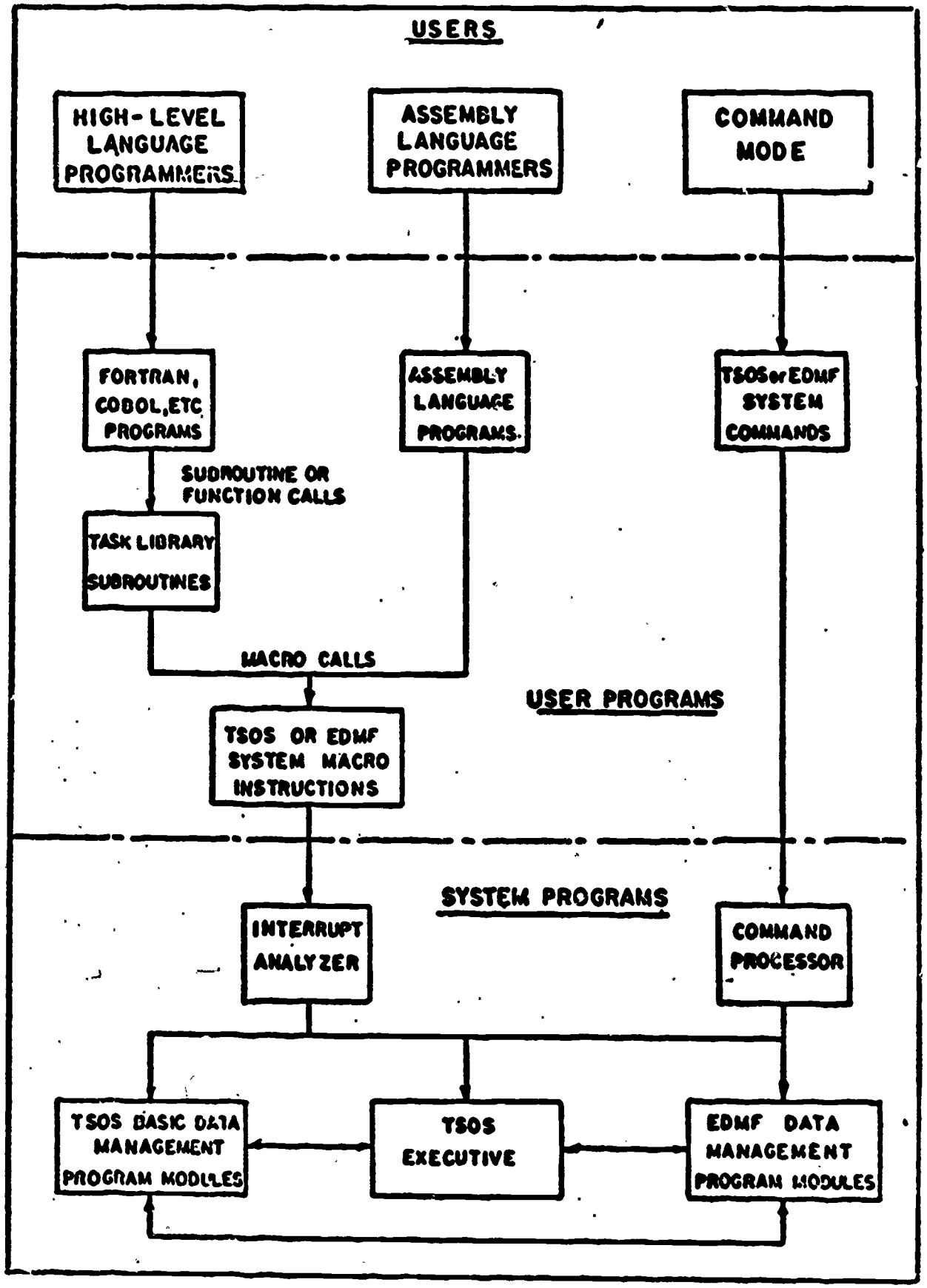


FIG. 4 EDMF - SOFTWARE ORGANIZATION

Work on logical access control mechanisms does not, of course, confined to the aforementioned references. Several recent software attempts at access control mechanisms can be found in [9, 10].

II. A LOGICAL ACCESS CONTROL SYSTEM

It is not possible to review in great details the complex logical access control mechanisms of PSF and EDMF in this short paper. For this reason we have attempted to present a simple system on which some of the important concepts and salient features of these mechanisms can be elaborated and from which an overall understanding of these mechanisms may be achieved.

Basically, the logical access control system consists of three parts:

(1) A shared data base to which access must be controlled,

(2) A group of users whose access to the data base must be regulated.

(3) A set of mechanisms which govern the accessing of the data base by the users.

In order to develop the system, solutions to the following problems must be provided. These solutions are discussed in separate sections.

(1) A method to identify the logical elements of the data base.

(2) A representation of the types of access the user can have to the data base.

(3) A concept on which a user can assign and change access types of other users to his data base.

(4) A methodology for effective implementations of various access control requirements as specified in the access types.

II.1 Identification of the Logical Elements of the Data Base

Logical elements in the data base must be uniquely identified so that the system can resolve their different identities and control access to individual elements. However, the burden of assigning unique identifiers to individual elements of data base should not be placed on the user. One requirement is therefore that the system, not the user, has the responsibility to assign a unique identification

number to each and every datum in the data base. What the user must have is a means to describe his logical elements for access control purpose and to specify his access control requirements for these elements. In order to describe his logical data elements for access control purpose, the user must be able to declare their structures; and in order to provide a priori and posterior controls over access to these data elements, the user must be able to specify their authentication and checking procedures. We note that the user already has a data description language for declaring data structure in data base creation and a data manipulation language for specifying data base management requirements. It follows that the same data description language may be used to declare the structure of the logical elements for access control purpose. The use of the same description language for both data base creation and control not only is expeditious but also eliminates the possibility of false identification due to ambiguity in language translation. Similarly, the same data manipulation language may be employed to specify the authentication and the checking procedure for access control to data base elements. Thus, a second requirement is that the means employed to describe elements of the data base for access control purposes and to specify their access control requirements should be the same as the ones used to declare data structure and to specify data manipulation requirements for data base creation and processing.

These are indeed the requirements that both PSF and EDMF attempted to meet. In both these systems elements of the data base are identified as fields, group of fields, keywords, records and files. References to fields are made by field names, to groups by group names and relations of the fields, to record by selection criterion in terms of Boolean and arithmetic expressions of fields and field names, and to files by file names. All these names and expressions can be used in the procedures for data definition, manipulation and access control purposes. In PSF, procedures can be written in either data description and manipulation command languages or data management assembly language macros. In EDMF higher-

level languages such as FORTRAN and COBOL have been extended to include data definition manipulation and access control facilities.

II.2 Representation of Access Types

We represent the types of access a user has to the data base as an authority item.

Conceptually, there is one-to-one correspondence between the users and the authority items.

An authority item consists of the following:

1. User identification
2. Names of all his accessible files
3. Options regarding the use of the files. Some of the options are listed below:

ND	Directory reading is not allowed
RR	Read records only
RE	Read either directory or records
RW	Read or write records
TB	Temporarily block the access of other users while the file is being modified
OWN	Own the file

4. Program entries for file-level procedural authentications.
5. Record access control descriptions (i.e., Boolean and arithmetic expression of keywords, field names and fields)

   For opening portions of files,

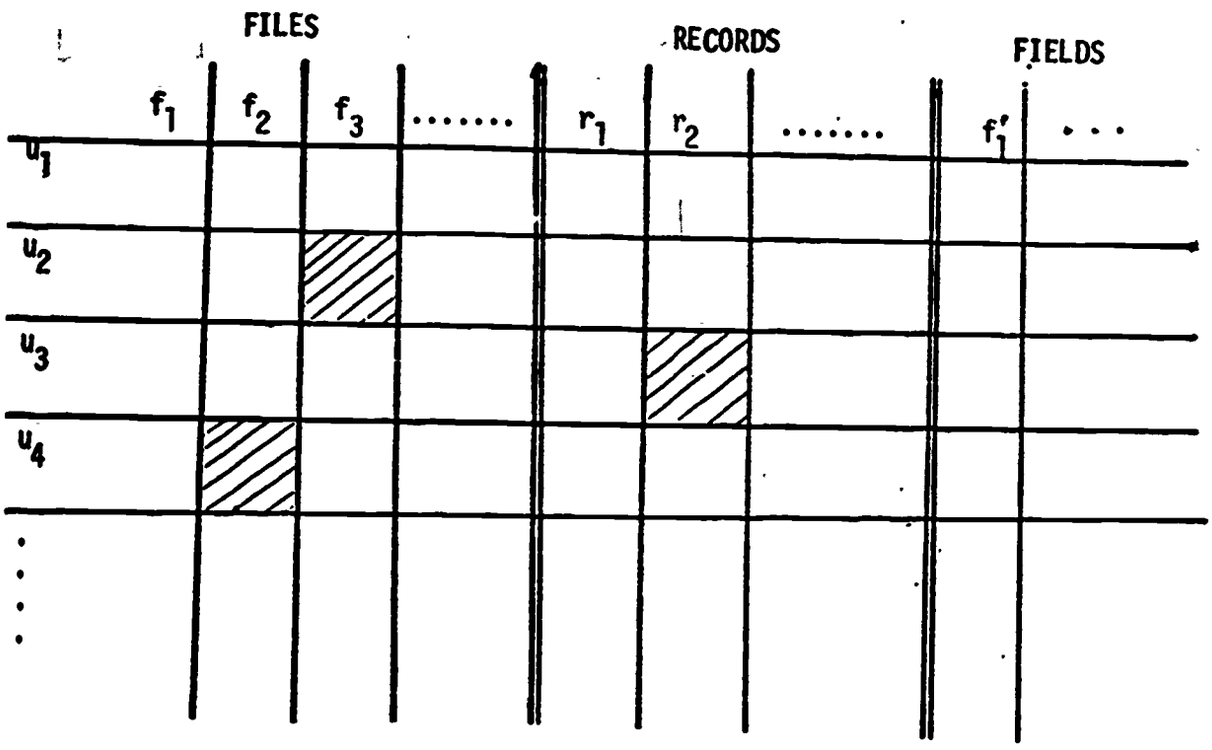
   For being temporarily blocked from access to portions of files,

   For allowing or denying ones use of certain portions of files.
6. Program entries for record-level procedural checking.

7. Field access control descriptions (i.e., Boolean and arithmetic expression of field and field names)

8. Program entries for field-level procedural checking.

The entire collection of the authority items may be viewed as an access control matrix with the user identified with the rows and logical units of data base the columns. The entry A_{uv} contains a series of access privileges and restrictions held by user u to logical unit v.



For implementation the matrix is too sparse to be stored as it is . For ease of change and update of access privileges and restrictions, the matrix should be organized as any other records of a (system) file. Because access privileges and restrictions to the same data units differ from one user to another, and because there are more data types than users, the implementation of authority items as records is user oriented instead of data type oriented. In other words,

there is one authority item for a user.

II.3 The Concept of Ownership

We introduce the concept of ownership to facilitate for the user the granting and denying of access privileges.

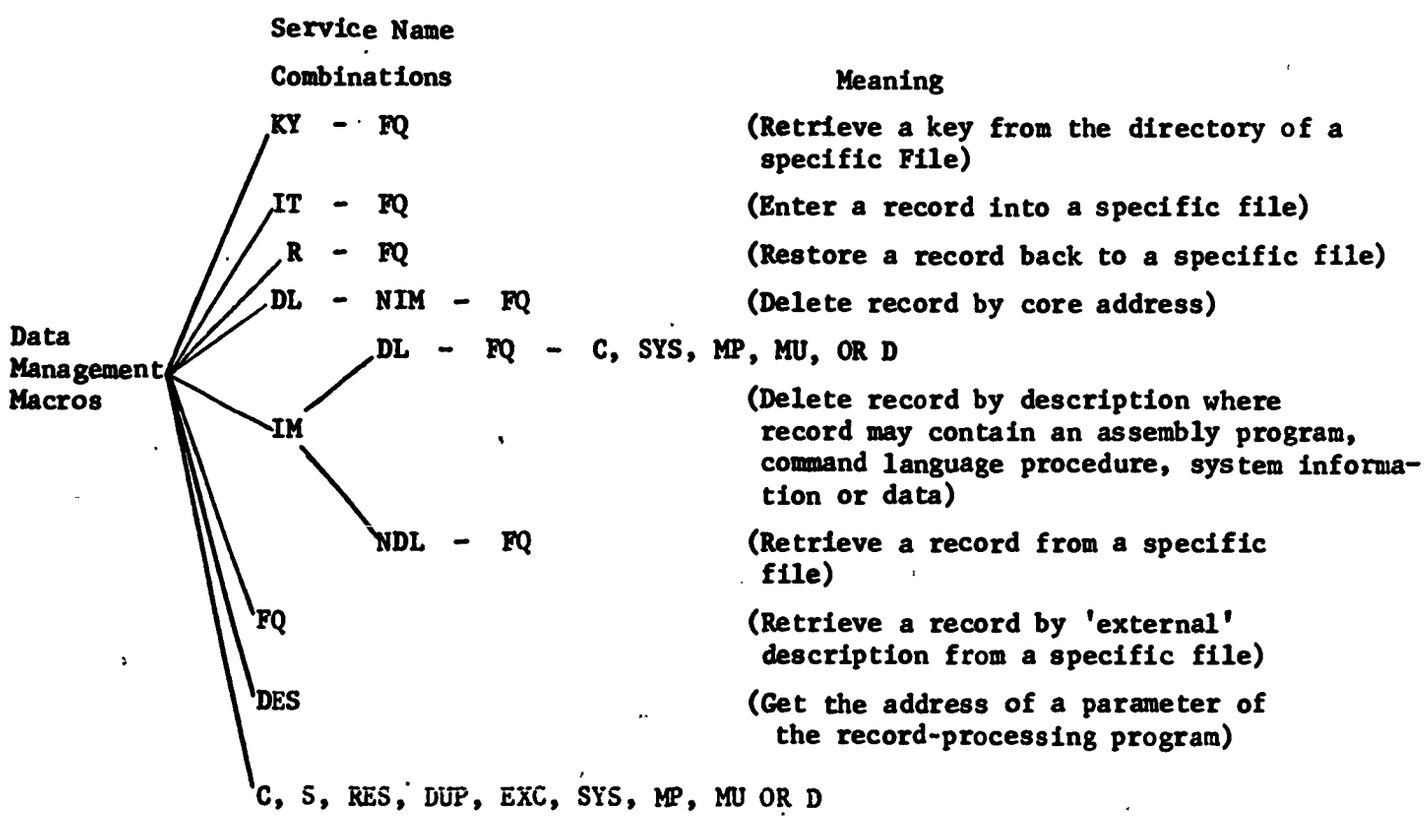
A user can grant and deny other users access to a file if he is the owner of the file.

The system administrator is a universal owner. Since he owns the file of authority items, he effectively owns every file in the system. To become an owner of a file the user must satisfy and comply with a set of rules and regulations which may vary from one installation to another. When a user becomes an owner of a file, the system assigns an OWN option for the file in the user's authority item. In the present implementations, the creator of a file will automatically become the owner of the file. As an owner of the file, the user can grant and deny any other users access to that file. In other words, a user with OWN option on a file can assign any option to other users for the file.

In particular, he may cause another user to be a co-owner of the file by assigning OWN option to that user.

The use of data management commands and macros by a user is dictated by the user's assigned options in his authority items.

Options to be provided to the system by the user's authority item to permit use of the command	Some of the Commands
None	New File
Own	Delete File
RR,RE,RW or OWN	Open File
None	Close File
Own	Access File
Own	No-Access File
TB or Own	Temporarily Block File
None	Unblock File
RW or Own	Reorganize File
None	Log-in for a File
Own	Enter a log-in Program



A typical service call has the following form:

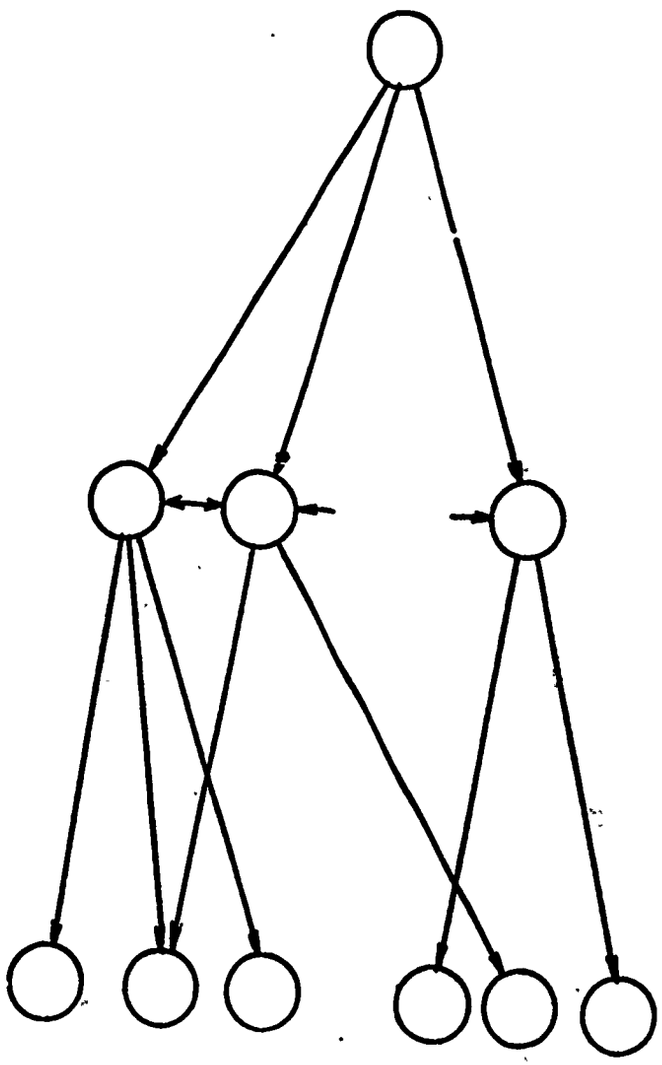
GSR P,(S_1, S_2, S_3,...,S_n)

Where GSR is the macro name, S_i are service names, and P indicates additional parameter involved with services. For example, the call

GSR ,(IT, FQ)

will enter a record from core (address in ac register) to the file named in mq register.

In summary, the concept of ownership establishes for the system a hierarchy of users as follows:



System Administrator
(1) Owns all files,
(2) Can use all data management commands and macros,
(3) May authorize other users to use his files,
(4) Is subjected to further rules and regulations.

File Owners
(1) Own private files,
(2) Can use all data management commands and macros,
(3) May authorize other users to use their files.

Non-Owner Users
(1) Do not own the files,
(2) Can use some data management commands and macros,
(3) May not authorize others to use the files.

II.4 Effective Implementations

Regardless the type of access the user may desire, there is a security procedure through which all access to the data base must be validated and monitored.

1. Identify the command or macro call
2. Check to see whether the user has access to the files involved
3. If user has access to a file, then check to see whether the file is currently open for his use.
4. If the file is open, then check to see if user has the proper file-option with respect to the call (e.g., read-only option for retrieval, write option for input, etc.).
5. If a user has the correct option for using the file, set up certain necessary information for the system programs involved.
6. Call the proper system program or programs to perform the requested service.
7. Keep track of the status of the service. Since a service may not be completed without repeated calls, it is necessary to save some information for the continuation of the service at a later time.
8. Update and save the information that was originally set up for the system programs.
9. Continue the service on next open file, if step 3 involves more than one open file.
10. Make sure that a record to be outputted is one belonging to the open portion of a file, not temporarily blocked from use by others, and not permanently protected from access.
11. Satisfy the procedural checking at record level.
12. Make certain that fields protected from access are removed from the outputting record.
13. Satisfy the procedural checking at field level.

Let us discuss some of the steps in the security procedure.

Steps 1 and 4 require the system to identify the types of options which are necessary for the issuance of the particular command or macro call. As was indicated in previous sections, the data management commands and macros that a user can exercise are determined by the type of options assigned to him. For example, without the OWN option the user is not permitted to delete a file through the issuance of the data management command DELETE FILE or the macro call IM-DL-FQ (with a Boolean expression characterized all records in the file). To this end, the system consults the user's authority item. Since options regarding a file are always associated with the name of the file as a part of the third entry in the authority item, the system can determine whether the user has proper options for the use of that command or macro call.

For step 2, the system consults the second entry of the authority item. We note that in this entry the names of all the accessible files to the user are listed; but files which are not accessible to him do not have their names in his authority item. In other words, inaccessible files are completely transparent to the user.

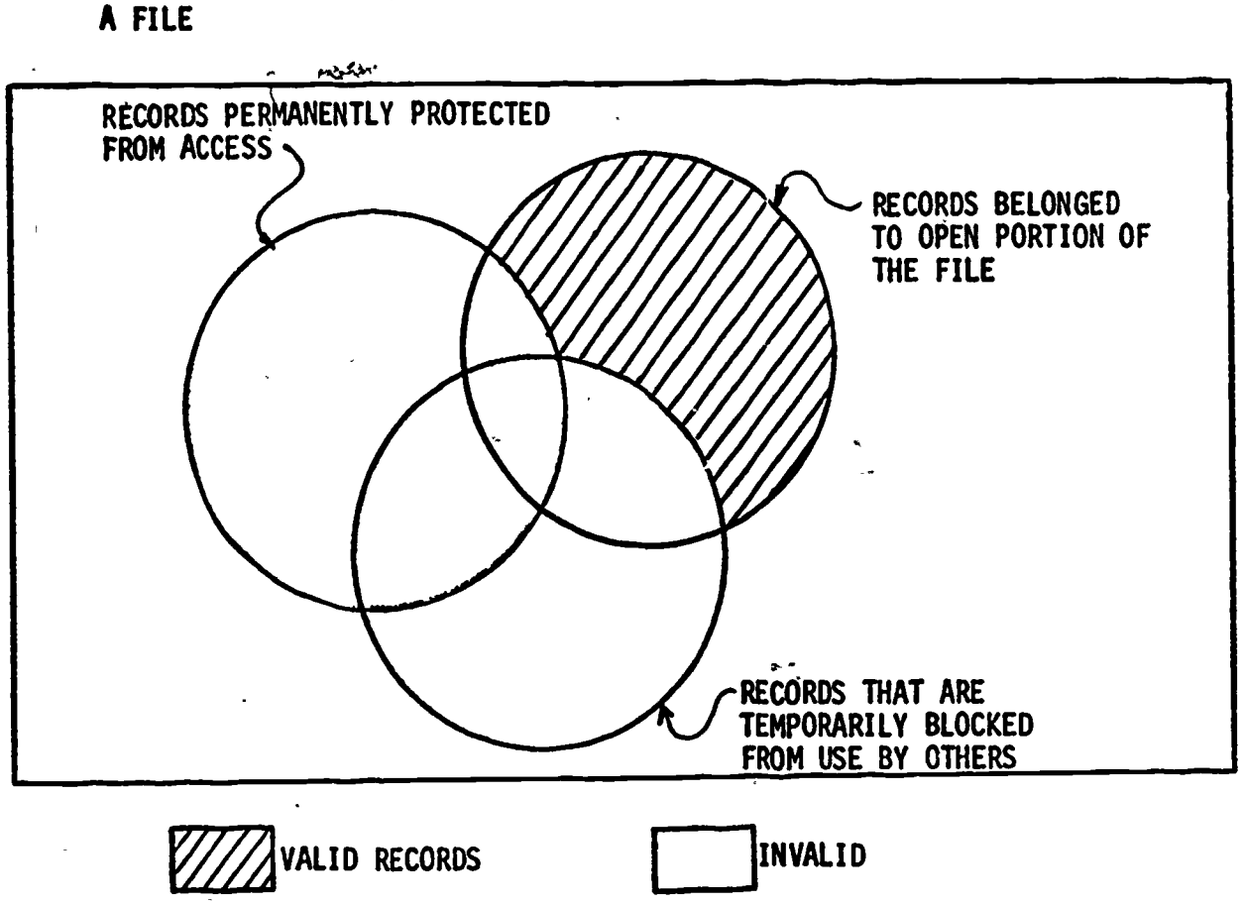
The inclusion of step 3 in the security procedure provides for a file owner the first opportunity to employ authentication programs to screen all users of the file. Typically an authentication program for a file consists of a set of user-written routines which are loaded into the system by the file owner through the use of the command ENTER-A-LOGIN-PROGRAM at the time of the file creation.

Although the program may demand various input from the user, it always produces either one of two standard output indicating whether there is either a positive or negative authentication. Since programs written for the authentication purpose can use data management macros, they can store and retrieve information regarding the number of file opening attempts and the combinations of user passwords.

We note that the program entry points for file-level checking are placed

in the fourth entry of the authority item. Thus, thoughtful use of the authentication programs (also known as log in procedure) at file level can provide very good protection of the file as a whole.

Steps 5 through 9 are self-explanatory. We shall not elaborate here. In step 10 the security procedure carries the access control form the file level as in step 3 to the subfile level. By using Boolean and arithmetic expressions of keywords, fields and field names as a means to partition a file into subfiles, the system can control access to the subfiles. However, in this case the partitions are virtual and no subfiles are actually being generated. The reasons for not physically making duplicate subfiles are to safeguard the integrity of the data base on the one hand and to facilitate update on the other hand. Virtual subfiles can be created readily by introducing new Boolean and arithmetic expressions. In fact, there can be a multiplicity of subfiles for various access control purposes as illustrated below.



THE SUBFILES OF RECORDS IN A FILE SPECIFIED BY THREE TYPES OF EXPRESSION.

On a need-to-know basis, many subfiles may be defined for the same files. The multiplicity of subfiles for various access control requirements may grow large. However, the mechanism needed by the system to verify whether a record belongs to a subfile is straightforward. Basically, the verification of a record with respect to an expression can be characterized by the following table.

Expression intended for...	Does the record satisfy the expression?	Validity of the Record
Permanent protection from use	Yes	Invalid
	No	Valid
Temporarily open for use	Yes	Valid
	No	Invalid
Temporarily blocked by others from use	Yes	Invalid
	No	Valid
⋮	⋮	⋮

From the above table, we note that an accessible record is a record which has been validated by every security check at the subfile level. These validations can be easily mechanized as depicted in Fig. 5.

In step 11, the system allows procedures incorporated by the file owner to check records which become accessible to a user. Whereas in step 3 access control is at the file level and in step 10 access control is at the subfile level, this step enables the control of access at record level. By allowing the file owner to develop his own record checking procedure, records which are already

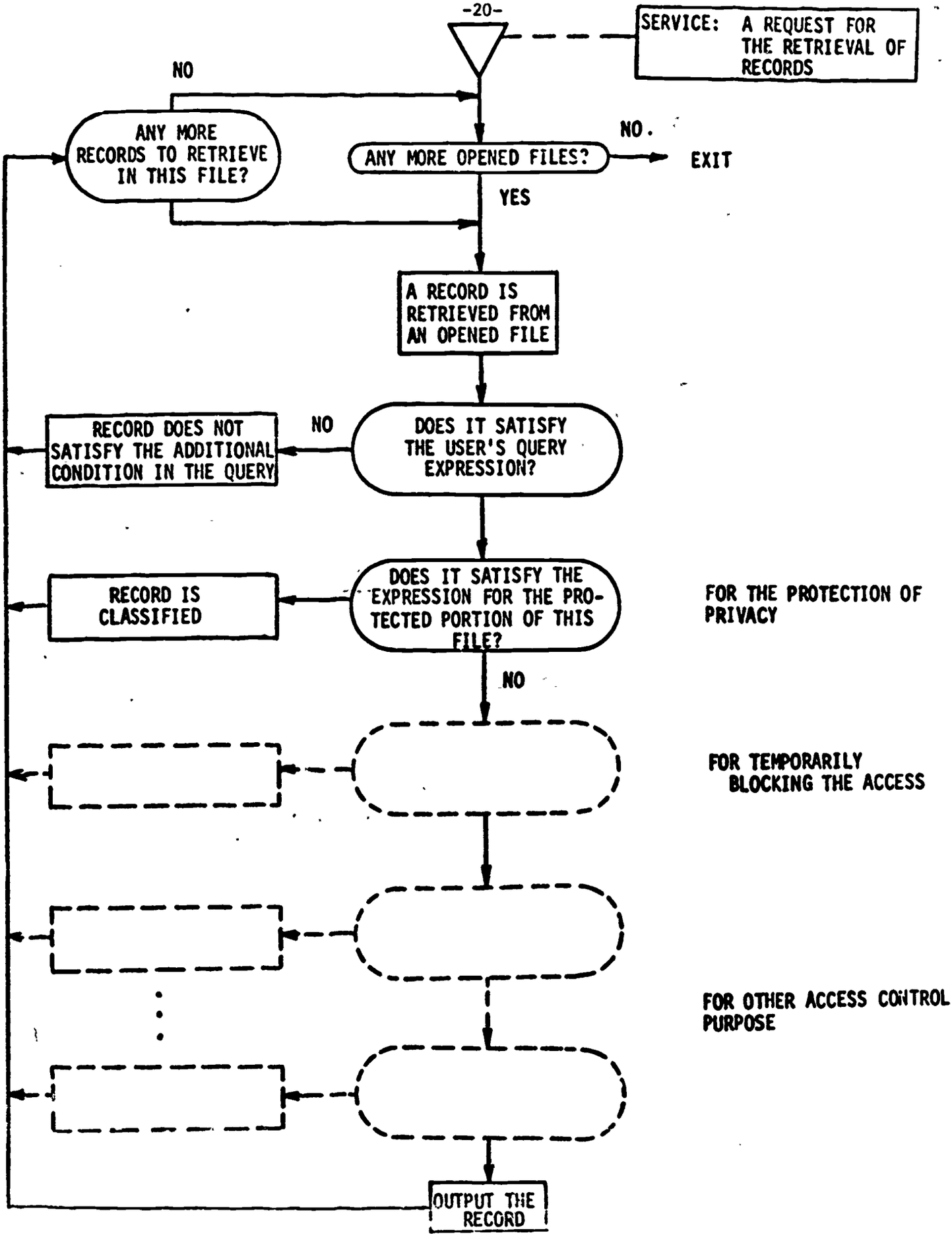


FIG. 5 EXPRESSION VALIDATIONS IN RECORD ACCESS

accessible to other users can be subject to further checking and auditing.

In step 12, the system performs posterior checking of protected field names. We note that the checking can only take place after the record has been retrieved from secondary storage into system's working area. In contrast to posterior checking, a priori checking of field names and fields at query time involves the removal of all the protected keywords, field names and fields from the user's data management commands and macro calls before the first step of the security procedure is to be invoked. In this way, no access will be initiated by any invalid keywords. Step 13 is the last step in the security procedure. It is in this step that access control is finally brought to the user at the field level. Because many access control requirements are dependent upon some combination of field names and/or values, matching of names and computing of values must take place dynamically. By incorporating these matching and computing procedures at the field level, the file owner can have direct control of other users access to his individual datum.

III. SUMMARY

We have discussed a simple logical access control system abstracted from the concepts and features of two experimental data base management systems known as PSF and EDMF. The highlights of the system are as follows:

1. Logical references for both data base access and access control - no reference to physical memory organization is made.
2. Flexible compartmentalization technique to aggregate information for access control - The aggregates are files, subfiles, records and fields. The use of Boolean and arithmetic expressions as a means to compartmentalize the files, records and fields eliminates the need of creating subfiles physically. Furthermore, expressions are stored in authority items and away from compartmentalized data. This procedure allows flexible change of compartments by simply undating the expressions.
3. Procedural approaches for further checking and authentication - At each logical level (i.e., file, subfile, record or field level) the file owner can introduce his own programs for auditing, monitoring and acreening other users of his data base.
4. Modular security procedure - The use of procedural checking and compartmentalization means by the user can vary. On the one hand, for public files there can be no compartmentalization of data within the files and no procedural checking at any level. On the other hand, multi-level checking on well-compartmentalized files can be demanded by the file owners. The security procedure is modular so that it can systematically invoke the right degree of security checking to support the user's security requirements.
5. Interlocking mechanism for blocking and unblocking multiple access to the same data aggregates - This mechanism is needed in an on-line environment where, for example, one user may update some data aggregates and another user may want to access the same data aggregates before the completion of

the update.

6. A priori and posterior control of field level access - A priori control is performed at assembly time of query statements. Posterior control is enforced at the time that an aggregate of data is about to be outputted to the user. The former is intended to prevent any access from taking place as a result of an invalid query; the latter is designed to remove small datum of highly classified nature from system output.

7. Ease of use - The introduction of the concept of ownership enables the users of the system to have an orderly way of obtaining and denying access privileges and restrictions. Because access control information is stored in the authority items, as records of a system file, it can be updated very easily by the system administrator. Furthermore, the separation of access control information from the raw data enables the change of a user's access requirements without having to process the raw data base.

REFERENCES

[1] Lampson, B.W., Scheduling and Protection in an Interactive Multi-Processor System. PhD. Dissertation, University of Calif. at Berkeley (March 1967).

[2] Schroeder, M.D., "Protection Mechanisms in MULTICS."

[3] Graham, R.M., "Protection in an Information Processing Utility," Comm. ACM 11, 5 (May 1968) pp. 365-369.

[4] Hsiao, D.K., A File System for a Problem Solving Facility (May 1968) NTIS AD 671 826.

[5] Hsiao, D.K., "Access Control in an On-Line File System," File Organization-Selected papers from File 68, IFIP ADP Group, Swets and Zeitlinger, Amsterdam, 1969.

[6] Hoffman, L.J., The Formulary Model for Access Control and Privacy in Computer Systems (May 1970) Ph.D. Dissertation, Stanford University.

[7] Manola, F.A., An Extended Data Management Facility for a General-Purpose Time-sharing System (May 1971) NTIS AD-724 801.

[8] Hsiao, D.K. and F.A. Manola, "Data Management with Variable Structure and Rapid Access," Proc. of USA-Japan Computer Conference (Oct. 1972) Tokyo, Japan.

[9] Owens, R., "Evaluation of Access Authorization Characteristics of Derived Data sets," 1971 ACM-SIGFIDET Workshop on Data Description, Access and Control,

[10] Conway, R., W. Maxwell, and H. Morgan, "Selective Security Capabilities in ASAP - A File Management System," AFIPS, 1972 SJCC, 40, pp. 1181-1185.

[11] Gelblat, M., "Computers for Medical Problem Solving - A New Medical Environment," Ph.D.Dissertation, The Moore School of Electrical Engineering, University of Pennsylvania, (Dec. 1971).