University of Pennsylvania
THE MOORE SCHOOL OF ELECTRICAL ENGINEERING

55

PROBLEMS IN PROTECTION OF INFORMATION IN A MULTIUSER ON-LINE SYSTEM

Jesus Arturo Ramirez

A thesis submitted to the Faculty of The Moore School of Electrical Engineering in partial fulfillment of the requirements for the degree of Master of Science in Engineering (for graduate work in Computer and Information Sciences).

Philadelphia, Pennsylvania

May, 1968

University of Pennsylvania
THE MOORE SCHOOL OF ELECTRICAL ENGINEERING

PROBLEMS IN PROTECTION OF INFORMATION IN A MULTI-USER ON-LINE SYSTEM

Abstract

The file system for the Moore School Problem Solving Facility of the University of Pennsylvania provides multiple files for various users and allows users to store, retrieve and update information in the files.

The working environment when using the file system is on-line, multiple-access and real time. In this kind of environment there exists problems which do not arise in a batch process or in a single-access on-line environment.

One of the problems that arises in using a file system in such an environment is that whenever a user makes changes in a particular file, he may cause other users, who are simultaneously accessing the same file to obtain erroneous information.

The objective of this thesis is to provide a solution to the above problem.

Master of Science in Engineering

(for graduate work in Computer and Information Sciences)

May, 1968

Signed:

J. A. Ramirez

Jesus Arturo Ramirez
Author

Noah Prywes

Dr. Noah S. Prywes
Faculty Supervisor

ACKNOWLEDGEMENT

The author wishes to thank Dr. Noah S. Prywes who suggested the subject of this work and whose encouragement and suggestions have been most helpful.

Thanks are also due to the entire staff of the Computer Center and to the members of Project Multilist, particularly to David Hsiao.

TABLE OF CONTENTS

		Page
List of Figures		v
Chapter 1	Introduction	1
Chapter 2	Description of the Protecting Features	4
	2.1 Interface Function with an On-Line System	4
	2.2 Authority-Item and Log-Table Structure	6
	2.3 The Blocking Function in Operation	14
Chapter 3	Functional Descriptions with Examples and Demonstrations	20
	3.1 The Block Function	20
	3.2 The Unblock Function	21
	3.3 Demonstration	22
	Case 1	24
	Case 2	27
	Case 3	34
	Case 4	37
Conclusions		41
References		42
Appendix		43

LIST OF FIGURES

		Page
Figure 1.1	Equipment Configuration of the Problem Solving Facility	2
Figure 2.1	Information Flow During the Processing of a MULTILANG Statement	5
Figure 2.2	An Authority-Item Format Specification	7
Figure 2.3	A Pointer to a Description	9
Figure 2.4	Last Word for a Data Block	10
Figure 2.5	Log-Table Format Specification	12
Figure 2.6	An Authority-Item As It Looks in Core	15
Figure 2.7	A Log-Table As It Looks in Core	17
Figure A-1	'Delete' Routine to Delete Copy of Translate Blocking Description from Log-Table	44
Figure A-2	'EXTRACT' Routine to Extract Description From Data Blocks in Authority-Item	45
Figure A-3	'FNFILE' Routine to Find Number of File Names In An Authority-Item	46
Figure A-4	'CNTKY' Routine for Counting the Number of Keys In A Non-Formatted Description	47
Figure A-5	'FNAME' Routine to Find a File Name in An Authority-Item	48
Figure A-6	'INSERT' Routine to Insert Description Into a Data Block Corresponding to a Given File Name In An Authority-Item	49
Figure A-7	The BLOCK Program	50
Figure A-8	The UNBLOCK Program	54

CHAPTER 1

INTRODUCTION

A computer system known as Moore School Problem Solving Facility (MSPSF)[1] has been designed and implemented using a central computer with a large memory, a smaller computer for input/output, and consoles for man-machine communication. The system consists of three parts:

a) An information storage and retrieval component, capable of storing large amounts of information including data and programs.

b) An Executive program that permits several users to communicate with the computer at the same time, sequencing input, execution, and output to provide real-time processing.

c) An Executive Language enabling a user to communicate requests for information and for program execution to the system.

Figure 1.1 shows the equipment used in the problem solving facility.

A file system has been added to the MSPSF which can serve not only as a repository for files of public, semiprivate and private information but also can provide users with functions for creating, searching, updating, reorganizing and purging files as well as automatically allocating storage and generating information of the files.

The following problem arises: In such an environment when a user of the file system is updating or reorganizing and purging of files, the changes may cause other users, who are accessing tne same file at that time, to obtain erroneous information.

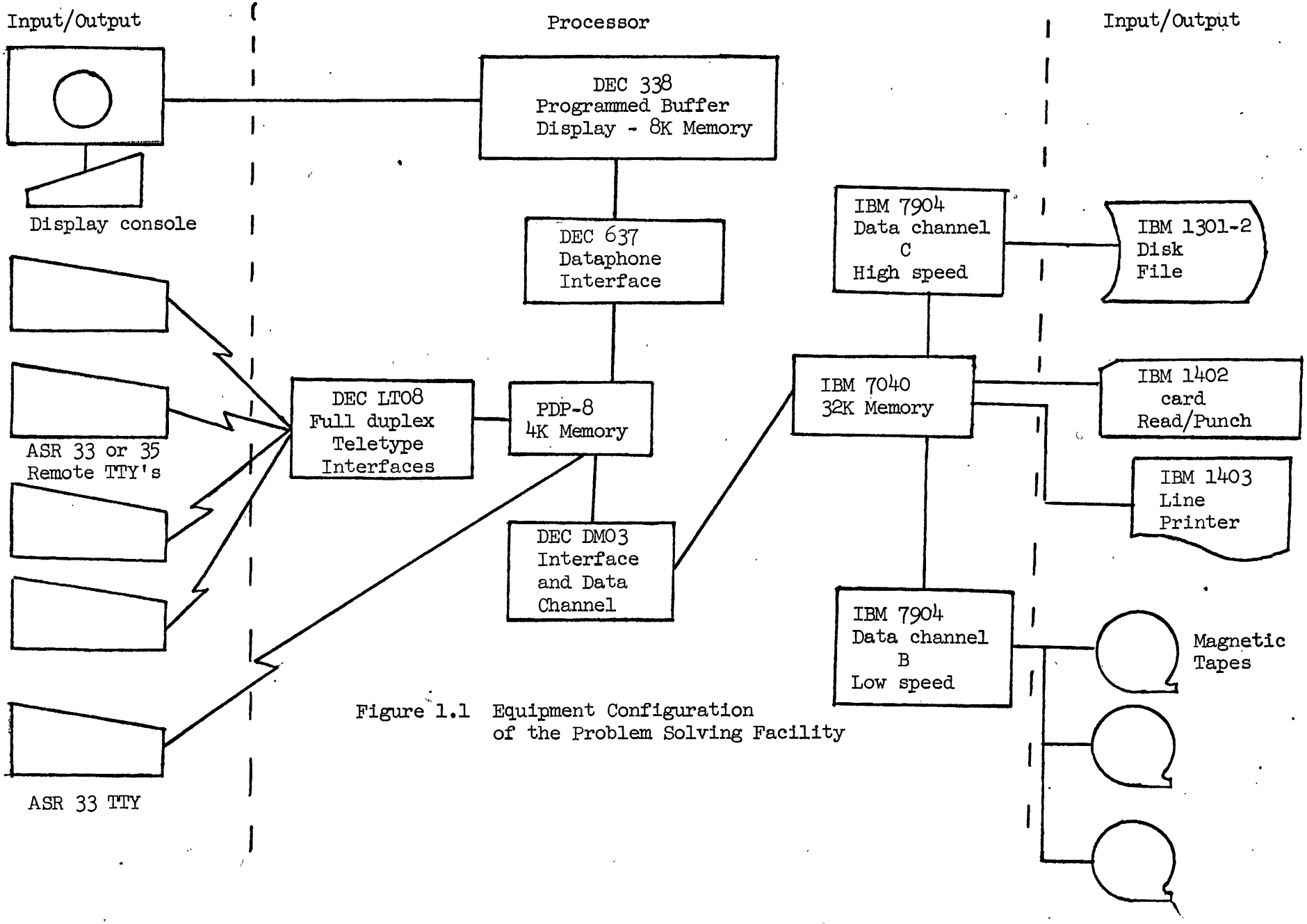


Figure 1.1 Equipment Configuration
of the Problem Solving Facility

One of the possible solutions that we devised was the following: provide the file system with a function which will enable a user to temporarily block others, otherwise eligible users, from accessing a file. Also, add to the file system another function which allows a user to unblock files which he previously blocked.

The Block and Unblock functions were designed taking into consideration that the system is an on-line, time sharing a number of users. The user can request the execution of Block or Unblock thru Multilang statements - namely through use of the same high level system language used for other system functions.

CHAPTER 2

DESCRIPTION OF THE PROTECTING FEATURES

2.1 Interface Function With An On-Line System

The systems Programs within the MSPSF are divided into two main subsystems: the Executive System and the Information Storage and Retrieval System (SRS). The former accepts statements in an Executive language, MULTILANG, in the format shown below:

XFILES OPERATION/OPERAND1/OP2/etc.

one pair

The operation is a description of the program to be executed (in this case Block or Unblock). The operands, when the program in question is Block, are used in pairs where the first element of the pair may be a file name or a disjunction of file names, the second one contains the information as to whether the user wants to block the entire file or only a portion (subset) of it.

When the program in question is Unblock, the operands are null since the function unblocks all the files previously blocked by the user requesting its execution.

Figure 2.1 shows the information flow during the processing of a MULTILANG statement. The statement is received by the Executive from the editing program of the PDP-8 and is translated to an internal format. The description of the desired 'operation' is located and tne key or set of keys for retrieval are selected.(2)

The SRS program is then used to retrieve from the file system (where the Block and Unblock programs are stored) the program having the specified retrieval key, then the Executive allocates storage for it

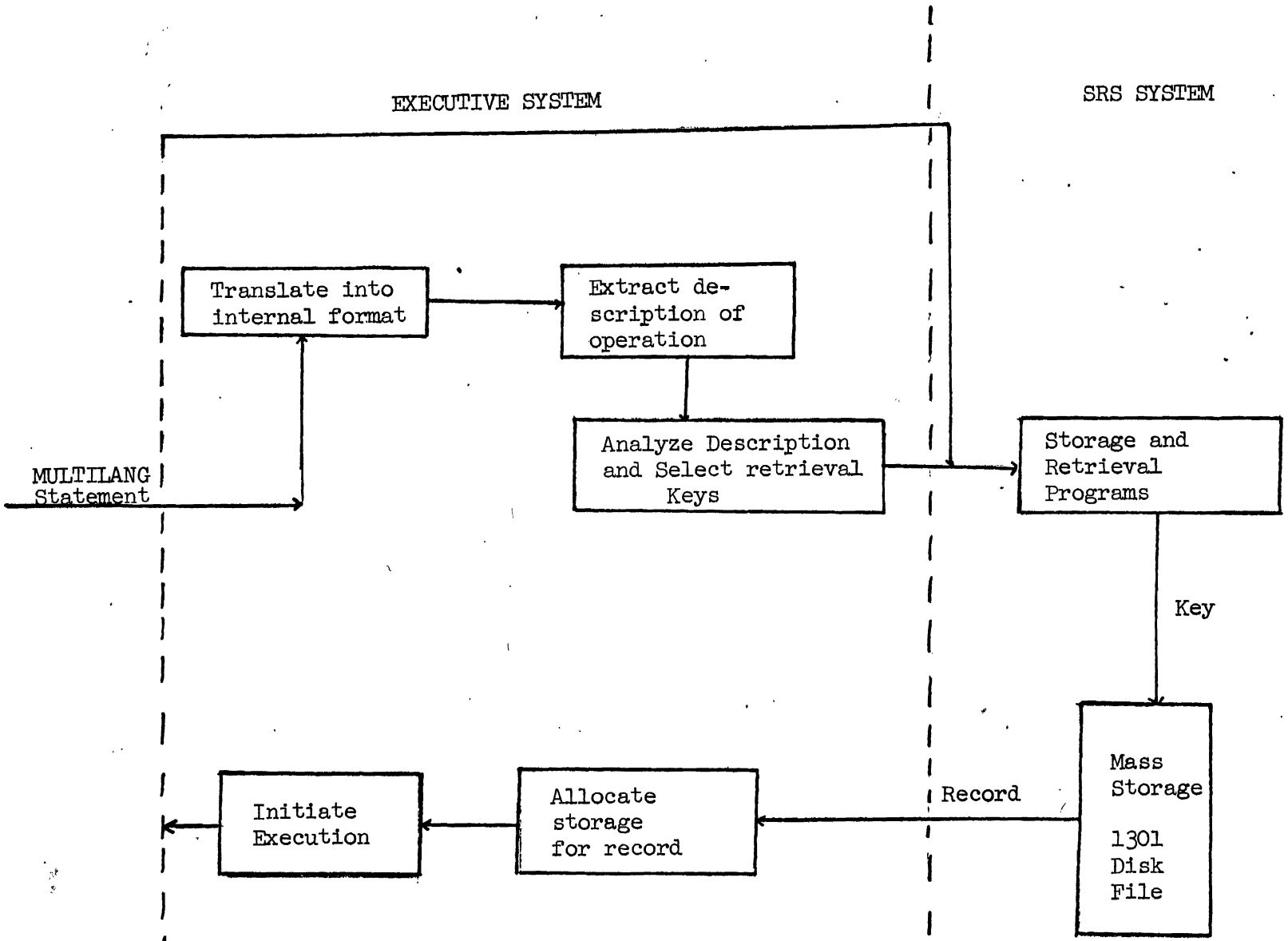


Figure 2.1 Information Flow During the Processing of a MULTILANG Statement

and initiates execution. At this point Block or Unblock has control and the function proceeds to perform the request. When this is done an appropriate set of messages is given to the user and control is passed to the system monitor.

2.2 Authority-Item and Log-Table Structure

At this point we introduce a little more detail about the basic units of information used by the Block and Unblock functions; these are the Authority-item and the Log-Table.

Generally speaking the authority-item contains information regarding the files which a user owns, the other files to which he has access, and the options for the use of these files.(3)

An Authority-item (see Fig. 2.2) is a block of consecutive storage locations which is divided into two sections -- the Control Section and the Data Section.

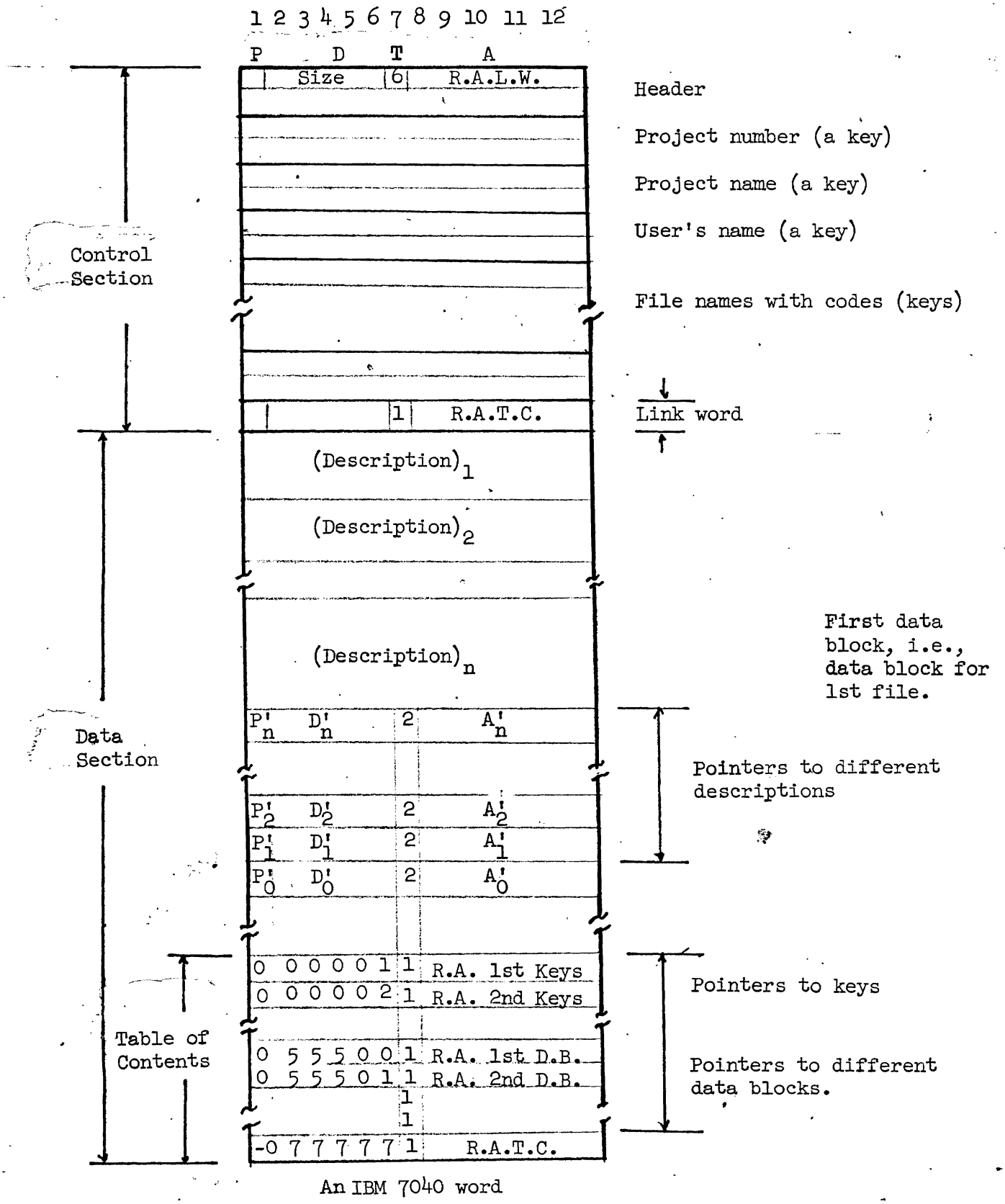
The Control Section carries the information needed to store an item initially, to retrieve it, and to restore it after it has been modified. In the Control Section of each authority-item all keys associated with it must appear. The keys are placed in alternate words beginning in word three.

The Data Section consists of two main sub-units:

a) a block of variable length for each file name to which a user has access, where the information and its pointers regarding the files are kept;

b) A block of variable length called the table of contents containing entries for each sub-unit of the authority-item.

An authority-item
format specification



An IBM 7040 word

Figure 2.2

Definitions:

Size: The number of words of the authority-item. It must be less than 463.

R.A.L.W.: Relative address (to the item) of the link word.

Key: Always takes two words.

Link word: The word between keys and data portions of the item. Always contains the relative address (to the item) of the Table of Contents.

R.A.T.C.: Relative address (to the item) of Table of Contents.

Data Block: A block of words consists of descriptions and their pointers.

Description: As far as the item is concerned, description is a number of words of variable size.

A_i^j: Relative address (to its data block) of the i-th description (i = 1, ..., n) in j-th data block.

D_i^j: See Fig. 2.3.

A_o^j: Relative address (to the data block) of the first pointer in j-th data block.

D_o^j: See Fig. 2.4.

R.A. j-th D.B. (with 55500 + j -1 in the decrement):
Relative address (to the item) of the last word of the j-th data block, i.e., relative address of the word containing D_o^j 2 A_o^j.

Note: The design of the authority-item takes the consideration of the following two points:

1) minimize the efforts in updating pointers;

2) retain the same basic format as data item.

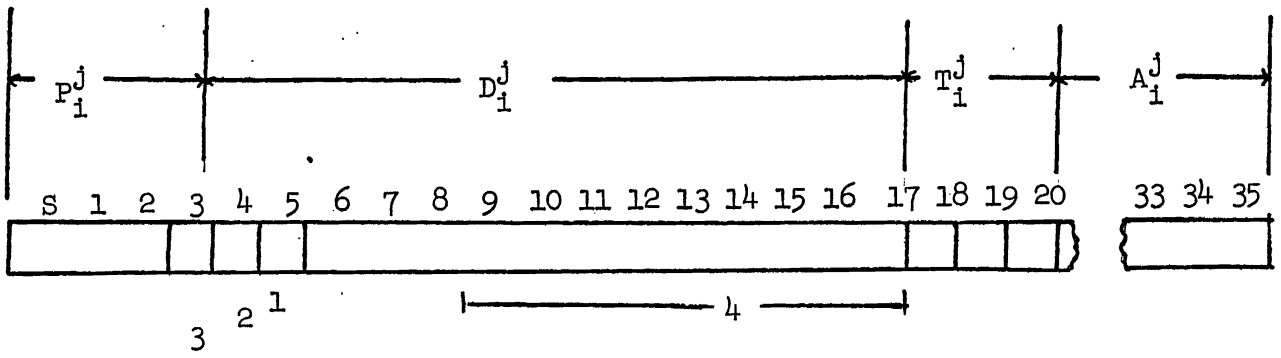


Figure 2.3 A Pointer to a Description

1. Access-bit: The description is used for allowing access to portion of the file of the j-th file.

2 BLOCK-bit: The description pointed by A_i^j is used for preventing the user from using records so described in j-th file.

3 OPEN-bit: The description pointed by A_i^j is used for opening portion of the file of the j-th accessible file.

4 When the Block-bit is set, this portion of the word is used to save the console number of the user who is being blocking j-th file.

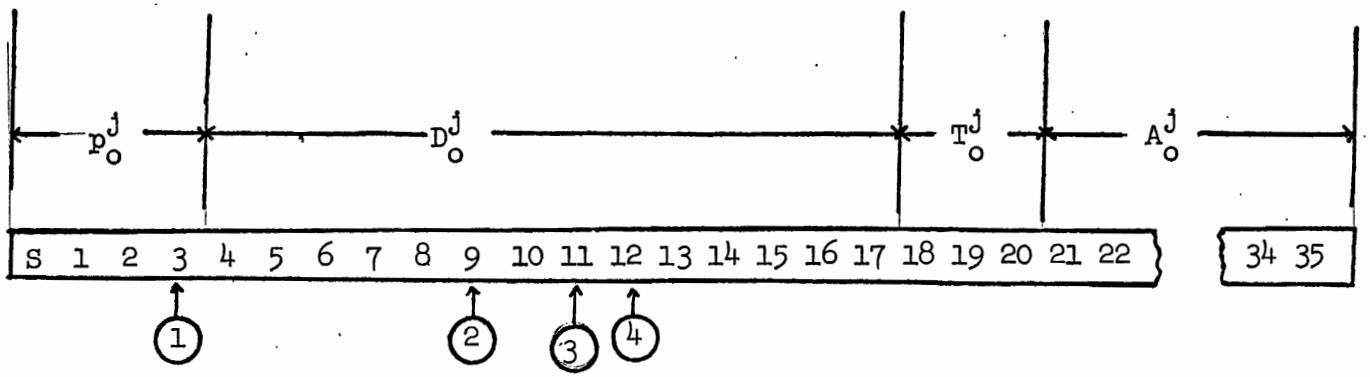


Figure 2.4 Last Word for a Data Block

1	OWN-bit:	Ownership Option
2	TB-bit:	Temporary-Blocking-Others Option
3	HTB-bit:	Having-Exercised-The-TB-Option Indicator
4	BTB-bit:	Being-Blocked-From-Using-File Indicator

A_o^j: Points to the first pointer in the j-th data block. In case when there are not descriptions A_o^j points to itself, i.e., the address part of the word is zeros.

The sub-units are called elements which are referred to by an associated number in the range 0 to 32767. This number is found in the decrement portion of the word in the Table of Contents associated with the element. The address portion of this word contains the relative address of the element within the authority-item.

The Log-Table (see Fig. 2.5) was designed to keep track of who is using the File System along with their status (blocked or unblock).

Similar to an Authority-item the Log-Table is divided into two sections -- the Control Section and the Data Section.

The first four words of the Log-Table make up the Control Section. There the information needed to store the Log-Table initially, retrieve it, and restore it after modification, is maintained.

The Data Section of the Log-Table consists of several sub-units:

a) A block of 20 consecutive words immediately after the link word, used to keep the user 'identification', 4 words per user (we need 20 words since at the moment the File system can handle a maximum of 5 consoles or terminals).

The first 4 words within the block correspond to a user using console No. 5, the next four to a user using No. 4, etc.

b) A block of 6 words immediately after the one used for user identification. The first word of this block carries the maximum number of users allowed for the file system, at this point 5. The remaining five words are used to keep the status of the users (as to whether they have been blocked or not). This is done in the following way: the decrement part of the word (the bits 3-17 which give room for 5 octal characters) and the 5 words make a square array of 5x5. Here the following con-

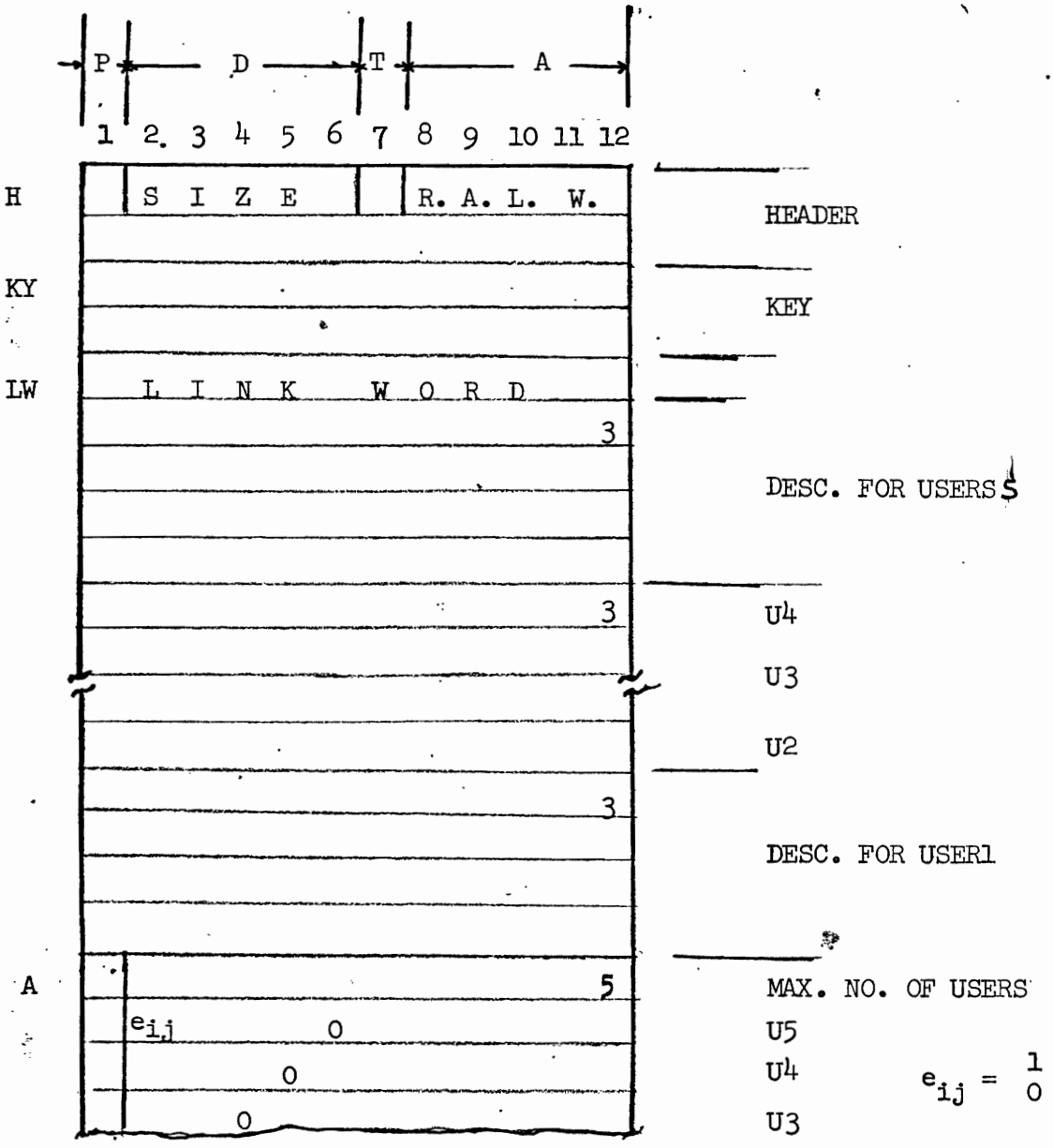


Figure 2.5 Log-Table Format Specification

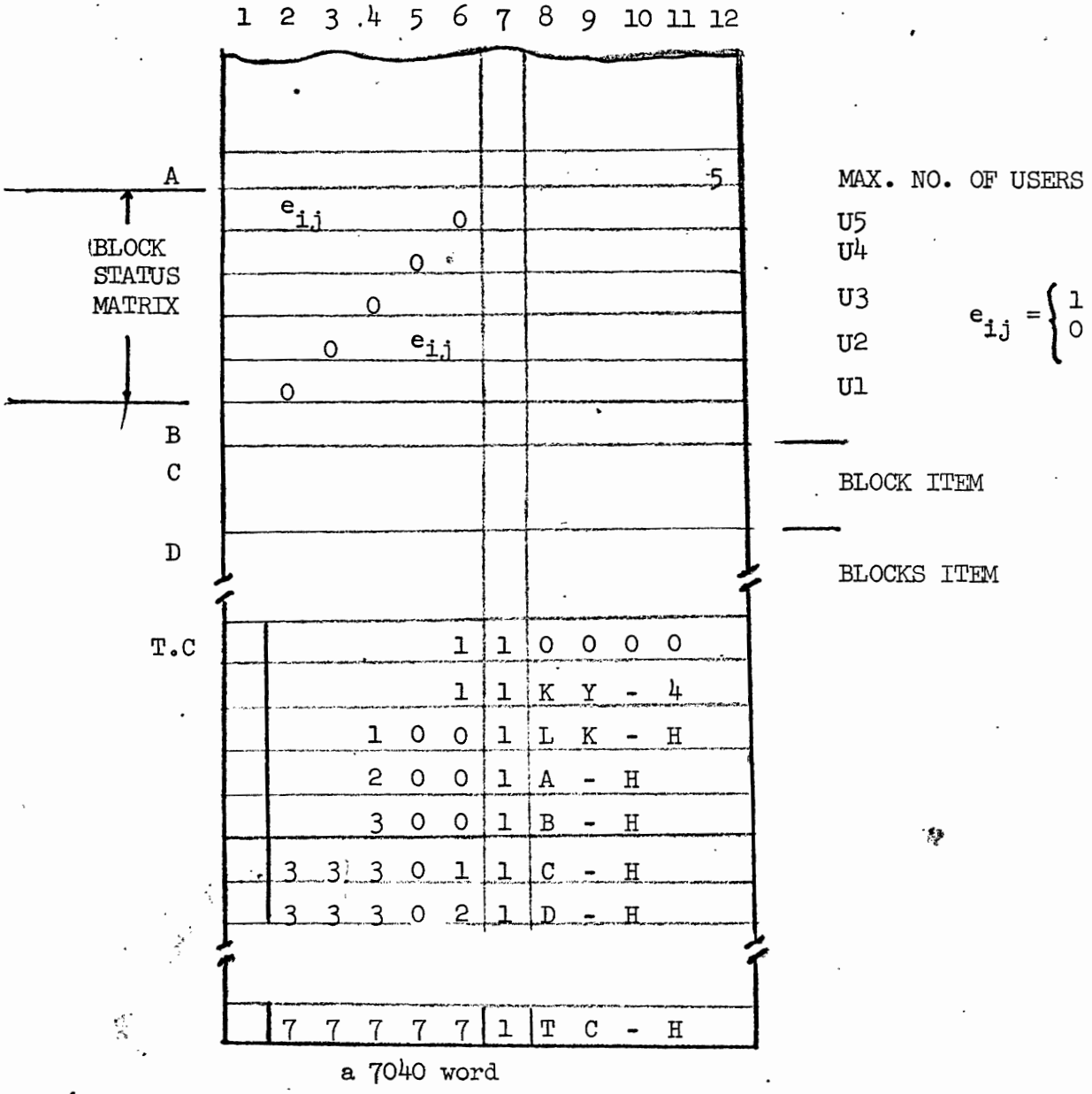


Figure 2.5 Log-Table Format Specification
(continued)

vention was adopted, if a e_{ij} is 1 then the user using console i was blocked by the user using console j, where $i \in \{1,2,3,4,5\}$ and $j \in \{1,2,3,4,5\}$.

c) One block of variable length (for each user who requested the use of the Block function), where a copy of the translated Multilang statement is kept.

d) A block of variable length called the Table of Contents containing entries for each sub-unit in the Log-Table.

Figure 2.6 shows how an authority-item looks in core, in this example the user's name is RAMIREZ, the Project name is MULTILIST and the Project number is 146077. The user owns 6 files and the right to block any of them.

Figure 2.7 shows how the Log-Table looks in core when 3 users are using the file system. They are HSIAO, RAMIREZ and FLINT using CONSOLES number 5, 4 and 3 respectively. At this point none of the users have blocked any of the files.

2.3 The Blocking Function in Operation

When a user requests the blocking of a file, the only indication that has to be made, as far as his authority-item is concerned, is in the last word of the data block corresponding to the file. There the HTB bit is set to indicating that he has temporarily blocked other users.

Referring to Fig. 2.6 suppose that this user blocks the MINSKY file to others, therefore the last word after this is done looks like:

before 043000200000

after 043200200000 ↑ MINSKY'S Data Block

On the other hand the information inserted in the Authority-Item of the other users using the system is as follows:

0	000073000040	'HEADER'
1	000005000233	
2	005121443151	USER NAME RAMIR
3	000002000225	
4	004464436331	PROJ. NAME MULTI
5	000004000233	
6	000104060007	PROJ. NUMBER 14607(7)
7	000004000233	
10	000103080207	PROJ. NO. 13827
11	000004000233	
12	000000010601	PROJ. NO. 00161
13	000004000233	
14	000000510107	BIN NUMBER R17
15	000004000233	
16	000000107051	BIN NUMBER 17R
17	000004000233	
20	000047642243	FILE 00PUBL
21	000004000233	
22	000100627062	FILE 010SYS
23	000004000233	
24	000244314562	FILE 02MINS
25	000004000233	
26	000362252162	FILE 03SEAS
27	000004000233	
30	000444252431	FILE 04MEDI
31	000004000233	
32	000547307062	FILE 05PHYS
33	000004000233	
34	000000100052	LINK WORD

Figure 2.6 An Authority-Item As It Looks in Core

35	-112144315125	RAMIRE
36	-316060606060	Z
37	-046443633143	MULTIL
40	316263606060	IST
41	010406000707	146077
42	010308020707	138277
43	000001060103	001613
44	043000200000	00PUBL's Data Block
45	043000200000	01OSYS's Data Block
46	043000200000	02MINS's Data Block
47	043000200000	03SEAS's Data Block
50	043000200000	04MEDI's Data Block
51	043000200000	05PHYS's Data Block
52	100001100002	Relative address of 1st KEY
53	100002100020	" " " 1st FILE NAME
54	100101100034	" " " Link word
55	100102100035	" " " extension of the Key
56	100103100036	" " " " " " "
57	100104100037	" " " " " " "
60	100105100040	" " " " " " "
61	100106100041	" " " " " " "
62	100107100042	" " " " " " "
63	100110100043	" " " " " " "
64	155001100044	Pointer to 1st data block
65	155011100045	" " 2nd " "
66	155021100046	" " 3rd " "
67	155031100047	" " 4th " "
70	155041100050	" " 5th " "
71	155051100051	" " 6th " "
72	-077777100052	Pointer to Table of Contents

Figure 2.6 An Authority-Item As It Looks in Core
(continued)

0	000047700006	Header	
1	000012000170		
2	000000434627	LOG	
3	000000000000		
4	006321224325	TABLE	
5	000000000000		
6	000001000042	LINK WORD	
7	000000000003		TERMINAL 5
10	003062312146	HSIA0	
11	000103080207	13827	
12	004464436331	MULTI	
13	000000000003		TERMINAL 4
14	005121443151	RAMIR	
15	000104060707	14607	
16	004464436331	MULTI	
17	000000000003		TERMINAL 3
20	002643314563	FLINT	
21	000000010601	00161	
22	004464436331	MULTI	
23	-000000000003		TERMINAL 2
24	000000000000		
25	000000000000		
26	000000000000		
27	-000000000003		TERMINAL 1
30	000000000000		
31	000000000000		
32	000000000000		
33	000000000005	MAX. NO. OF USERS	
34	000000000000	U5	
35	000000000000	U4	
36	000000000000	U3	
37	000000000000	U2	
40	000000000000	U1	
41	000000000000		
42	000001100002		
43	000100100006		
44	000200100033		
45	000300100041		
46	-077777100042		

Figure 2.7 A Log-Table As It Looks in Core

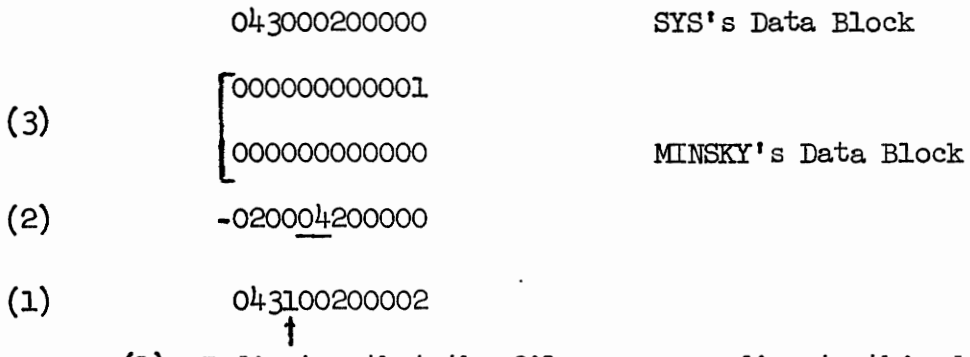
a) The BTB bit is set to 1 in the last word data block corresponding to Minsky file.

b) The necessary information to indicate that the Minsky file has been completely blocked is inserted in the MINSKY data block.

c) Next, the console number of the user who made the request is entered in the pointer to the blocking description and the BLOCK and Validity bits are set to 1.

d) The final step is the updating of all necessary points in such authority-items.

The figure below shows the modification:



(1) Indicates that the file corresponding to this data block has been blocked.

(2) Is the pointer to the blocking description where the validity and block bits are set to 1 and also entered the console number (04) of user who blocked the file.

(3) Is the blocking description, in this case indicates that the whole file is blocked.

The modifications made in the Log-Table are the following:

a) In row 2 of the Block Status matrix (i.e., the corresponding to user at console No. 4) bits 12 and 18 are set to one to indicate that users 3 and 5 have been blocked by user 4.

b) A copy of the translated Multilang statement is stored in Log-Table and a pointer to this request is added in the Table of Contents (suppose just for the sake of simplicity the size of this block is 10 octal words).

c) Finally all necessary updatings are performed.

000000000003	Max. no. of users	
000000000000	U5	
000101000000	U4	(a)
same as before		
000000000000		
copy of translated MULTI-LANG statement	10 octal words	(b)
same as before		
000300100041		
033304100042		(c)
-077777100053		

CHAPTER 3

FUNCTIONAL DESCRIPTIONS WITH EXAMPLES AND DEMONSTRATIONS

3.1 The Block Function

After a user (henceforth called the current user) has asked for the Block function through a Multilang statement and the system monitor has released control to the block function, the request is analyzed in order to see if it is well formed (i.e., as far as the syntax of the Executive language is concerned). Then the user's authority-item is retrieved from disk into core and a series of checks is performed. This checking consists of determining whether or not the files which he has requested be blocked exist in his authority-item, and if he has the privilege of blocking such files and finally if he himself has not been blocked by others from the use of some of these files. If one of these test results is positive, an appropriate message is given to the user.

The next step is the setting of information in the corresponding pointers to the files being blocked to indicate that the blocking is taking effect. Then the user's authority-item is restored to disk.

After this is done, the Block function proceeds to investigate which other users are presently using the file system. This is achieved by retrieving the Log-Table where such information is kept. By searching thru it, the Block function gets the information necessary to retrieve the authority-items of the other users, one by one. A test is done in order to see if each of these users has access to the files in question, if that is the case, the blocking is performed by inserting into his authority-item the blocking description and putting it back into disk. As a final step a copy of the Translated request (i.e., the Multilang statement) is stored into the Log-Table under the entry point corresponding

to the current user. Also, the corresponding bit is set up in the 'Block Status Matrix' to indicate that the user in question is being blocked by the current user.

The Block function is called by the system monitor whenever a new user enters the system, since some of the files to which he has access may previously have been blocked by other users.

The Block function searches thru the Table of Contents in the Log-Table for pointers to copies of translated Multilang "Block" statements entered by other users. Each one found is made effective with regard to the new user.

The entry to the Block program depends upon whether it is called by a user or by the system monitor. This is so because when the system monitor calls on Block there is no need to translate and copy a Multilang statement, nor to give any of the eligibility messages.

3.2 The Unblock Function

The Unblock function provides the user with the ability to unblock files that he previously blocked.

The unblocking of files previously blocked by a user (henceforth called current user) takes place in the following way:

After a user has asked for the Unblock function through a Multilang statement and the syntax of the request has been checked, his authority-item is retrieved, a search for files blocked by him is performed and if there are any the information is masked out and his authority-item restored to disk.

Otherwise, an appropriate message is given and control is passed to the system Monitor.

The next step performed is the retrieval of the Log-Table in order to find out which users were blocked by the current user. With this information in hand the authority-item of such a user is retrieved from disk and all the blocking information entered by the current user is deleted.

The user's authority-item is then restored to core. This procedure is carried on for each user who was blocked by the current user.

The Unblock function is called by the system monitor when a user leaves (i.e., 'signs-out') to delete from his authority-item any blocking descriptions entered there by other users. The monitor also calls on Unblock to unblock files which the current user previously blocked and has neglected to set free.

3.3 Demonstration

This part of the paper describes the performance of the Block and Unblock functions in various actual situations. We first give an explanation of the various messages which the system uses to communicate with the user.

To a request of the type:

XFILES BLOCK/F1/D1/F2/D2/etc.

the Block function types out a group of messages depending on the status of the File system.

a) XFILES BLOCK/F1/D1/F2/D2/etc.

END OF MESSAGES.

This indicates that the Block Command was satisfactorily executed.

b) XFILES BLOCK/F1/D1/F2/D2/etc.

YOU'VE BEEN BLOCKED BY OTHERS FOR THE USE OF THE FILE...F1

END OF MESSAGES.

This indicates that all the files with the exception of F1 have been blocked as requested.

c) XFILES BLOCK/F1/D1/F2/D2/etc.

USER DOES NOT HAVE THE RIGHT TO BLOCK THIS FILE

F2

END OF MESSAGES.

This message differs from b in that it indicates the non-availability of F2 to the user, and the whole request is ignored.

Let us now consider the messages typed out by the Unblock function. After a user has typed in

XFILES UNBLOCK

a) XFILES UNBLOCK

XFILES UNBLOCK HAS BEEN EXECUTED...

END OF MESSAGES.

The message is self-explanatory.

b) XFILES UNBLOCK

ALL FILES THAT USER WANTS TO

UNBLOCK HAVE NOT BLOCKED BY HIM

END OF MESSAGES.

This message too is self-explanatory.

Case 1: (see Page 25)

Here we examine the simplest case - that of two users, one at terminal A and the other at terminal B.

Suppose that both users request and receive permission to access the file system. Now if user A blocks the MINSKY file, in particular records of that file which contain the name GRAY in them, user B will be unable to access it. If user B now attempts to block the MINSKY file he cannot do so and the BLOCK function indicates this by the message.

XFILES BLOCK/MINSKY/GRAY

YOU'VE BEEN BLOCKED BY OTHERS FOR THE USE OF FILE .... MINS

BUT YOU MAY BLOCK OTHER FILES.

END OF MESSAGES.

Later the user at terminal A UNBLOCK'S the MINSKY file and now an attempt by B to block it will be successful.

```
C I@
C O@
A@  NO INPUT LINES
XUSERS/RAMIREZ/MULTILIST/138277
XFILES OPEN
START@
P O@
XUSERS/RAMIREZ/MULTILIST/138277
YOU MAY USE THE FILE SYSTEM.
XFILES OPEN
THE FOLLOWING FILES ARE OPENED TO YOU
       PUBL, ØSYS, MINS, SEAS, MEDI, PHYS,
END OF MESSAGES.
```

Terminal A
$t = t_0$

```
C I@
C O@
A@  NO INPUT LINES
XUSERS/FLINT/MULTILIST/138277
START@
P O@
XUSERS/FLINT/MULTILIST/138277
YOU MAY USE THE FILE SYSTEM.
```

Terminal B
$t = t_0$


```
C I@
C O@
A@  NO INPUT LINES
COUNT/GRAY/MINSKY
XFILES BLOCK/MINSKY/GRAY
START@
P O@
       COUNT/GRAY/MINSKY
ITEM COUNT OF FILE 'MINS' = 00001
      TOTAL COUNT  =  00001
XFILES BLOCK/MINSKY/GRAY
END OF MESSAGES.
```

Terminal A
$t = t_1$

```
C I@
C O@
A@  NO INPUT LINES
XFILES OPEN
START@
P O@
XFILES OPEN
THE FOLLOWING FILES ARE OPENED TO YOU
       PUBL, ØSYS, MINS, SEAS, MEDI, PHYS,
END OF MESSAGES.
```

Terminal B
$t = t_1$

Terminal A

doing other jobs,
involving files blocked
by him.

$t = t_2$

```
C I@
C O@
A@  NO INPUT LINES
XFILES BLOCK/MINSKY/GRAY
COUNT/GRAY/MINSKY
START@
P O@
XFILES BLOCK/MINSKY/GRAY
YOU'VE BEEN BLOCKED BY OTHERS FOR THE USE OF FILE...    MINS
BUT YOU MAY BLOCK OTHER FILES.
END OF MESSAGES.
        COUNT/GRAY/MINSKY
      TOTAL COUNT   =   00000
```

Terminal B
$t = t_2$

```
C I@
C O@
A@  NO INPUT LINES
XFILES UNBLOCK
START@
P O@
XFILES.UNBLOCK
XFILES UNBLOCK HAS BEEN EXECUTED....
END OF MESSAGES.
```

Terminal A
$t = t_3$

```
C I@
C O@
A@  NO INPUT LINES
XFILES BLOCK/MINSKY/GRAY
 COUNT/GRAY/MINSKT
START@
P O@
XFILES BLOCK/MINSKY/GRAY
END OF MESSAGES.
        COUNT/GRAY/MINSKT
ITEM COUNT OF FILE 'MINS' = 00001
      TOTAL COUNT   =   00001
```

Terminal B
$t = t_4$

Case 2: (see page 28)

In this case, 3 users request the use of the file system, namely RAMIREZ, HSIAO, FLINT; each one blocks the others from use of a file.

The blocking goes as follows: RAMIREZ blocks HSIAO and FLINT from use of file MEDICAL and in particular of records of that file which contain the name TROOP. HSIAO blocks RAMIREZ and FLINT from use of the file SEAS and in particular of records containing GORN in them. Finally, FLINT blocks RAMIREZ and HSIAO from use of the file MINSKY and in particular of records containing GRAY in them.

Later the three users try to block files previously blocked by others, then the Block function indicates this by corresponding messages.

As a next step the three users using the file system call UNBLOCK in order to unblock files previously blocked by them.

Finally to show that Unblock performs properly its mission each one of the users attempts again to block the files to which they were blocked before.

```
FULL@   CONSOLE AVAILABLE
SIGNIN@
C I@
C O@
A@    NO INPUT LINES
XUSERS/RAMIREZ/MULTILIST/146077
START@
P O@

XUSERS/RAMIREZ/MULTILIST/146077
 YOU MAY THE FILE SYSTEM.
```

Terminal A
$t = t_0$

```
FULL@  CONSOLE AVAILABLE
SIGNIN@
C I@
C O@
A@ NO INPUT LINES
XUSERS/HSIAO/MULTILIST/138277
START@
P O@

XUSERS/HSIAO/MULTILIST/138277
 YOU MAY USE THE FILE SYSTEM.
```

Terminal B
$t = t_0$

```
FULL@  CONSOLE AVAILABLE
SIGNIN@
C I@
C O@
A@ NO INPUT LINES
XUSERS/FLINT/MULTILIST/001613
START@
P O@

XUSERS/FLINT/MULTILIST/001613
 YOU MAY USE THE FILE SYSTEM.
```

Terminal C
$t = t_0$

```
C I@
C O@
A@  NO INPUT LINES
XFILES OPEN
START@
P O@

XFILES OPEN
THE FOLLOWING FILES ARE OPEN TO YOU
       PUBL, ØSYS, MINS, SEAS, MEDI, PHYS,
END OF MESSAGES.
```

Terminal A
$t = t_1$

```
C I@
C O@
A@  NO INPUT LINES
XFILES OPEN
START@
P O@

XFILES OPEN
THE FOLLOWING FILES ARE OPEN TO YOU
       PUBL, ØSYS, MINS, SEAS, MEDI, PHYS,
END OF MESSAGES.
```

Terminal B
$t = t_1$

```
C I@
C O@
A@  NO INPUT LINES
XFILES OPEN
START@
P O@

XFILES OPEN
THE FOLLOWING FILES ARE OPEN TO YOU
       PUBL, ØSYS, MINS, SEAS, MEDI, PHYS,
END OF MESSAGES.
```

Terminal C
$t = t_1$

```
C I@
C P@
A@  NO INPUT LINES
XFILES BLOCK/MEDICAL/TROOP
START@
P O@

XFILES BLOCK/MEDICAL/TROOP
END OF MESSAGES.
```

Terminal A
$t = t_2$

```
C I@
C O@
AP  NO INPUT LINES
XFILES BLOCK/SEAS/GORN
START@
P O@

XFILES BLOCK/SEAS/GORN
END OF MESSAGES.
```

Terminal B
$t = t_2$

```
C I@
C O@
A@  NO INPUT LINES
XFILES BLOCK/MINSKY/GRAY
START@
P O@

XFILES BLOCK/MINSKY/GRAY
END OF MESSAGES.
```

Terminal C
$t = t_2$

```
C I@
C O@
A@  NO INPUT LINES
XFILES BLOCK/SEAS/GORN/MINSKY/GRAY
START@
P O@

XFILES BLOCK/SEAS/GORN/MINSKY/GRAY
YOU'VE BEEN BLOCKED BY OTHERS FOR THE USE OF FILE...   SEAS
YOU'VE BEEN BLOCKED BY OTHERS FOR THE USE OF FILE...   MINS
BUT YOU MAY BLOCK OTHER FILES.
END OF MESSAGES.
```

Terminal A
$t = t_3$

```
C I@
C O@
A@  NO INPUT LINES
XFILES BLOCK/MEDICAL/TROOP/MINSKY/GRAY
START@
P O@

XFILES BLOCK/MEDICAL/TROOP/MINSKY/GRAY
YOU'VE BEEN BLOCKED BY OTHERS FOR THE USE OF FILE...   MEDI
YOU'VE BEEN BLOCKED BY OTHERS FOR THE USE OF FILE...   MINS
BUT YOU MAY BLOCK OTHER FILES.
END OF MESSAGES.
```

Terminal B $t = t_3$

```
C I@
C O@
A@  NO INPUT LINES
XFILES BLOCK/MEDICAL/TROOP/SEAS/GORN
START@
P O@

XFILES BLOCK/MEDICAL/TROOP/SEAS/GORN
YOU'VE BEEN BLOCKED BY OTHERS FOR THE USE OF FILE...   MEDI
YOU'VE BEEN BLOCKED BY OTHERS FOR THE USE OF FILE...   SEAS
BUT YOU MAY BLOCK OTHER FILES.
END OF MESSAGES.
```

Terminal C
$t = t_3$

```
C I@
C O@
A@  NO INPUT LINES
XFILES UNBLOCK
START@
P O@

XFILES UNBLOCK
XFILES UNBLOCK HAS BEEN EXECUTED.........
```

Terminal A
$t = t_4$

```
C I@
C O@
A@  NO INPUT LINES
XFILES UNBLOCK
START@
P O@

XFILES UNBLOCK
XFILES UNBLOCK HAS BEEN EXECUTED.........
```

Terminal B
$t = t_4$


```
C I@
C O@
A@  NO INPUT LINES
XFILES UNBLOCK
START@
P O@

XFILES UNBLOCK
XFILES UNBLOCK HAS BEEN EXECUTED.........
```

Terminal C
$t = t_4$

```
C I@
C O@
A@  NO INPUT LINES
XFILES BLOCK/SEAS/GORN
START@
P O@

XFILES BLOCK/SEAS/GORN
END OF MESSAGES.
```

Terminal A

$t = t_5$

```
C I@
C O@
A@  NO INPUT LINES
XFILES BLOCK/MINSKY/GRAY
START@
P O@


XFILES BLOCK/MINSKY/GRAY
END OF MESSAGES.
```

Terminal B

$t = t_5$

```
C I@
C O@
A@  NO INPUT LINES
XFILES BLOCK/MEDICAL/TROOP
START@
P OP

XFILES BLOCK/MEDICAL/TROOP
END OF MESSAGES.
```

Terminal C

$t = t_5$

Case 3: (see page 35)

In this case we are illustrating the situation in which a new user arrives to use the file system but at the same time other users using the system before him had blocked files to which this incoming user normally has access to.

The user who is using the system (say at t = 0) is RAMIREZ and he has requested the Blocking of the file MEDICAL and in particular of records in that file containing TROOP in them.

At $t = t_n$ the MEDICAL file is still blocked and the user HSIAO arrives to use the system, and since he normally has access to such file he will be blocked. Thus when he requests the use of the Block function in order to Block 'MEDICAL' the Blobk function gives him the corresponding message.

```
C I@
C O@
A@  NO INPUT LINES
XUSERS/RAMIREZ/MULTILISR/138277
START@
P O@
XUSERS/RAMIREZ/MULTILISR/138277
YOU MAY USE THE FILE SYSTEM.
```

Terminal A
$t = t_0$

```
C I@
C O@
A@  NO INPUT LINES
 COUNT/TROOP/MEDICAL
XFILES BLOCK/MEDICAL/TROOP
START@
P O@
   .    COUNT/TROOP/MEDICAL
ITEM COUNT OF FILE 'MEDI' = 00001
      TOTAL COUNT  =  00001
XFILES BLOCK/MEDICAL/TROOP
END OF MESSAGES.
```

Terminal B
$t = t_1$

Terminal B

Nobody is using this

terminal at the moment.

Terminal B
$t = t_2$

```
SIGNIN@

C I@
C O@
A@  NO INPUT LINES
XUSERS/HSIS-AO/MULTILIST/138277

START@
P O@
XUSERS/HSIAO/MULTILIST/138277
YOU MAY USE THE FILE SYSTEM.
```

User at terminal A

doing other jobs,

involving files blocked.

by him.

```
@

C I@
C O@
A@  NO INPUT LINES
XFILES OPEN
COUNT/GRAY/MINSKY
START&@
P O@
XFILES OPEN
THE FOLLOWING FILES ARE OPENED TO YOU
        PUBL, @SYS, MINS, SEAS, MEDI, PHYS,
END OF MESSAGES.
        COUNT/GRAY/MINSKY
ITEM COUNT OF FILE 'MINS' = 00001
      TOTAL COUNT   =  00001
```

Terminal B

$t = t_3$

```
C I@
C O@
A@  NO INPUT LINES
COUNT/TROOP/MEDICAL
XFILES BLOCK/MEDICAL/TROOP
STATC#
START@
P O@
       COUNT/TROOP/MEDICAL
      TOTAL COUNT   =  00000
XFILES BLOCK/MEDICAL/TROOP
YOU'VE BEEN BLOCKED BY OTHERS FOR THE USE OF FILE...    MEDI
BUT YOU MAY BLOCK OTHER FILES.
END OF MESSAGES.
```

Terminal B

$t = t_4$

Case 4: (see Page 38)

In this case we examine the situation which arises when a user, having blocked some files leaves the system, i.e., he 'signs-out' without having requested the Unblocking of such files.

The user who leaves the system is RAMIREZ and he had previously requested the Blocking of the files MINSKY and MEDICAL. FLINT is the other user.

Later Ramirez 'SIGNS-OUT'. Then the System monitor calls on Unblock and the Unblocking of such files is automatically taken care of.

To show that the unblock of MINSKY and MEDICAL took effect the user FLINT requests the Blocking of such files and this time he is successful.

```
FULL@  CONSOLE AVAILABLE
SIGNIN@
C I@
C O@
A@  NO INPUT LINES
XUSERS/RAMIREZ/MULTILISY-T/138277
XFILES BLOCK/MINS/GRAY
START@
P O@
XUSERS/RAMIREZ/MULTILIST/138277
YOU MAY USE THE FILE SYSTEM.
XFILES BLOCK/MINS/GRAY
END OF MESSAGES.
```

Terminal A
$t = t_1$

```
C I@
C O@
A@  NO INPUT LINES
 COUNT/TROOP/MEDICAL
XFILES BLOCK/MEDICAL/TROOP
START@
P O@
        COUNT/TROOP/MEDICAL
ITEM COUNT OF FILE 'MEDI' = 00001
      TOTAL COUNT  =  00001
XFILES BLOCK/MEDICAL/TROOP
END OF MESSAGES.
```

Terminal A
$t = t_2$

```
@FULL@  CONSOLE AVAILABLE
SIGNIN@
C I@  7040 IS NOT READY
C I@
C I@
C O@
A@. NO INPUT LINES
XUSERS/FLINT/MULTILIST/138277
XFILES OPEN
START@
P O@
XUSERS/FLINT/MULTILIST/138277
YOU MAY USE THE FILE SYSTEM..
XFILES OPEN
THE FOLLOWING FILES ARE OPENED TO YOU
        PUBL, 0SYS, MINS, SEAS, MEDI, PHYS,
END OF MESSAGES.
```

Terminal B
$t = t_0$

User at terminal B doing other jobs.

User at terminal A

doing other jobs related

to files blocked by him.

```
C I@
C O@
A@  NO INPUT LINES
COUNT/TROOP/MEDICAL
XFILES BLOCK/MEDICAL/TROOP
START@
P O@
       COUNT/TROOP/MEDICAL
      TOTAL COUNT   =   00000
XFILES BLOCK/MEDICAL/TROOP
YOU'VE BEEN BLOCKED BY OTHERS FOR THE USE OF FILE...     MEDI
BUT YOU MAY BLOCK OTHER FILES.
END OF MESSAGES.
```

Terminal B
at $t = t_3$

```
C I@
C O@
A@  NO INPUT LINES
COUNT/GRAY/MINSKY
XFILES BLOCK/MINSKY/GRAY
STARTT@
P O@
       COUNT/GRAY/MINSKY
      TOTAL COUNT   =   00000
XFILES BLOCK/MINSKY/GRAY
YOU'VE BEEN BLOCKED BY OTHERS FOR THE USE OF FILE...     MINS
BUT YOU MAY BLOCK OTHER FILES.
END OF MESSAGES.
```

Terminal B at $t = t_3$

```
C I@
C O@
A@  NO INPUT LINES
XSIGNOUT
START@
P O@
XSIGNOUT
YOU HAVE SIGNED OUT.
```

Terminal A
at $t = t_4$

```
C I@
C O@
A@  NO INPUT LINES
XFILES BLOCK/MINSKY/GRAY
 COUNT/GRAY/MINSKY
START@
P O@
XFILES BLOCK/MINSKY/GRAY
END OF MESSAGES.
       COUNT/GRAY/MINSKY
ITEM COUNT OF FILE 'MINS' = 00001
      TOTAL COUNT  =  00001
```

Terminal B
at $t = t_5$

```
C I@
C O@
A@  NO INPUT LINES
 COUNT/TROOP/MEDICAL
XFILES BLOCK/MEDICAL/TROOP
START@
P O@
       COUNT/TROOP/MEDICAL
ITEM COUNT OF FILE 'MEDI' = 00001
      TOTAL COUNT  =  00001
XFILES BLOCK/MEDICAL/TROOP
END OF MESSAGES.
```

Terminal B at $t = t_6$

CONCLUSIONS

As stated in the introduction the aim of this paper is the design and implementation of a set of functions which will prevent several different users of an on-line, time-sharing system, from accessing the same file simultaneously. As we showed in the previous chapter, the Block and Unblock functions are capable of solving the problems arising when a user is working with a file and has requested that other users be temporarily blocked from using the file. The problem of preventing a new user from accessing a file, which he can normally access and another user has blocked, and the problem of assuring that all files, that a user has blocked before, are unblocked when he leaves the system, have been successfully solved.

It is also important to note that we took advantage of the ability of the system to grow. As a result, we were able to make the interaction of the Block and Unblock functions and the existing system simple and efficient.

REFERENCES

(1) Wexelblat, R., "The Development and Mechanization of a Problem Solving Facility," Ph.D. Dissertation, University of Pennsylvania, December 1965.

(2) Morton, R.P. and Wolfberg, M.S., "The Input/Output and Control System of the Moore School Problem Solving Facility," Report No. 67-30, The Moore School of Electrical Engineering, University of Pennsylvania, June 1967.

(3) Hsiao, D., "A File System for a Problem Solving Facility," Ph.D. Dissertation, University of Pennsylvania, May 1968.

APPENDIX

Routine used for deleting the copy of the translate blocking description from Log-Table.

NAME: DELETE

INPUT: 1) Address of Log-Table.
2) Address of block's pointer where blocking desc is.

OUTPUT: DELETION of translated copy of Multilang Statement.

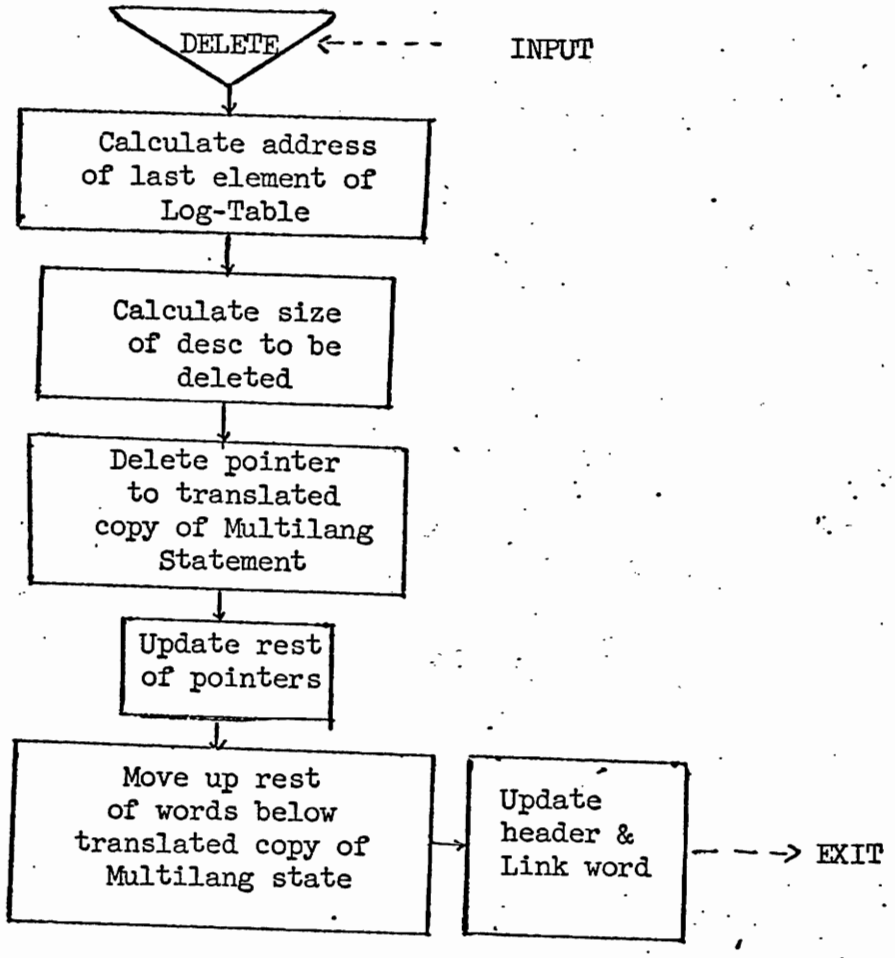


Figure A-1
'Delete' Routine to Delete Copy of Translate Blocking Description from Log-Table

NAME: EXTRAC: Routine for extract description from data blocks in auth-items

INPUT: 1) Comp. of auth-item's header in XR1.

2) Comp. of address of pointer corresponding to file from where we want extrac descriptions.

3) Rel addr (with respect to first word in data block) of pointer to description in 'ACC'

4) Size of auth-item in 'MQ'

OUTPUT: 1) The description will be extracted and all pointers will be updated.

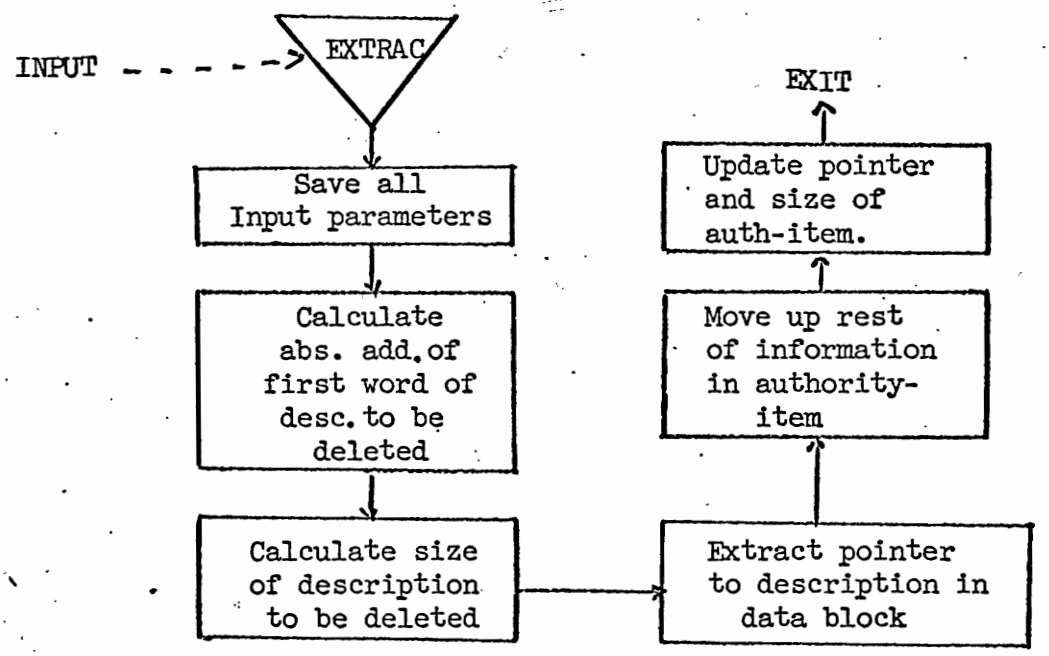


Figure A-2
'EXTRACT' Routine to Extract Description from Data Blocks in Authority Item

NAME: FNFILE

INPUT: Complement of address of header of auth-item in ACC

OUTPUT: 1) Number of files in auth-item in ACC.

2) Rel. addr. of link word in XR1

3) Comp. of address of f. file in XR2

4) Comp. of address of T. of C. in XR3

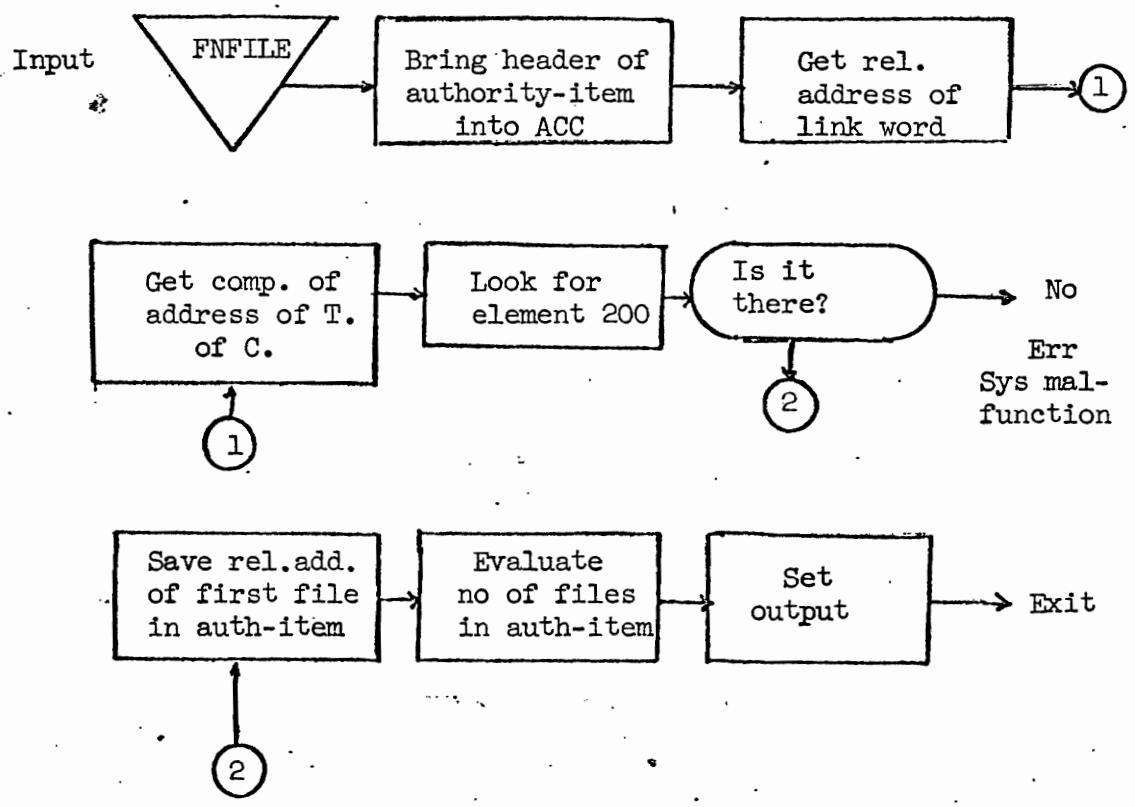


Figure A-3
'FNFILE' Routine to Find Number of File Names
in an Authority-Item

NAME: CNTKY: Routine for counting the number of keys in a non-formatted description.

INPUT: The complement of the address of the Description in XR1.

OUTPUT: The no. of keys in the description in XR2.

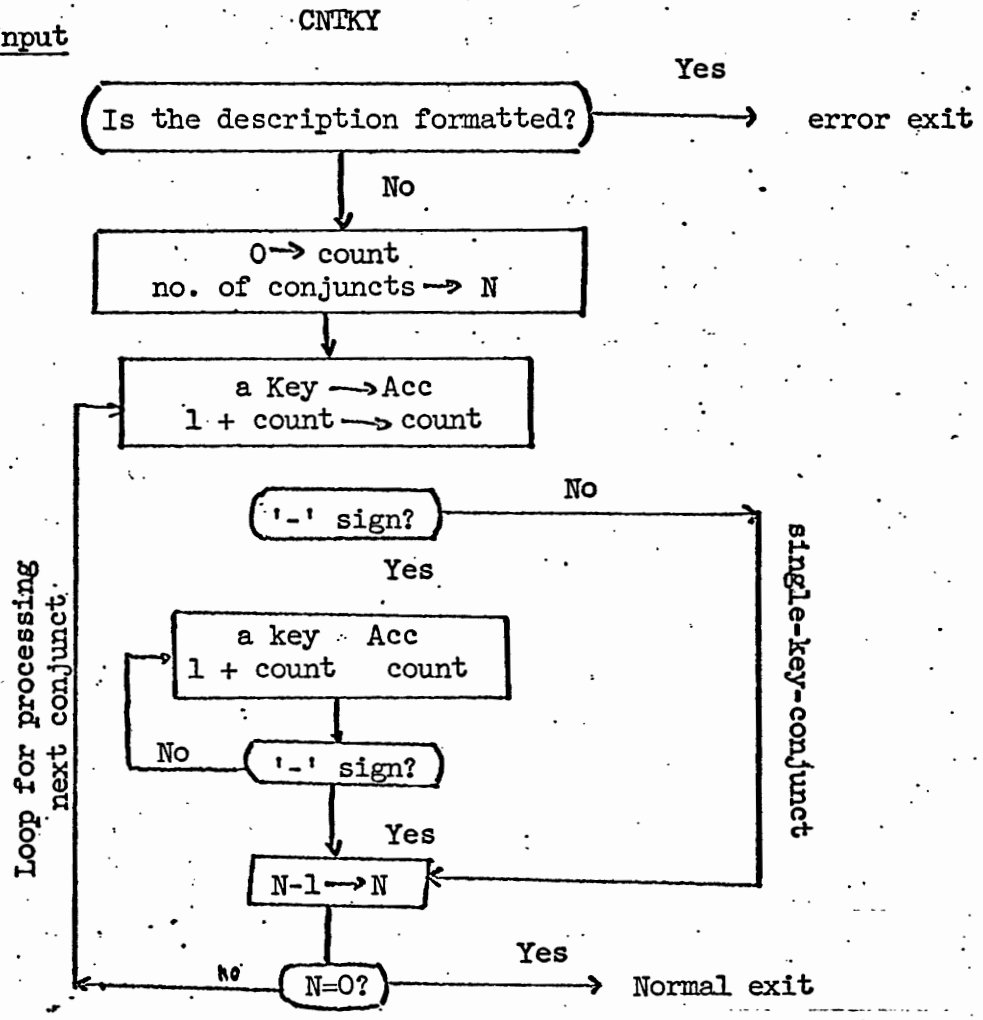


Figure A-4
'CNTKY' Routine for Counting the Number of Keys in a Non-Formatted Description

NAME: FNAME

INPUT: 1) File name that will be compared with auth-item's f. names in MQ.

2) No. of files names in auth-item in ACC.

3) Comp of address of first f. name in XR1

4) " " " " T. of C. in XR4.

OUTPUT: 1) The "P" bit of accumulator is set to '1' if the f. name in question doesn't match with someone in auth-item.

2) If the file name is in auth-item and its corresponding pointer is in T of C the complement of its address is put into ACC, otherwise the ACC is set to zeros.

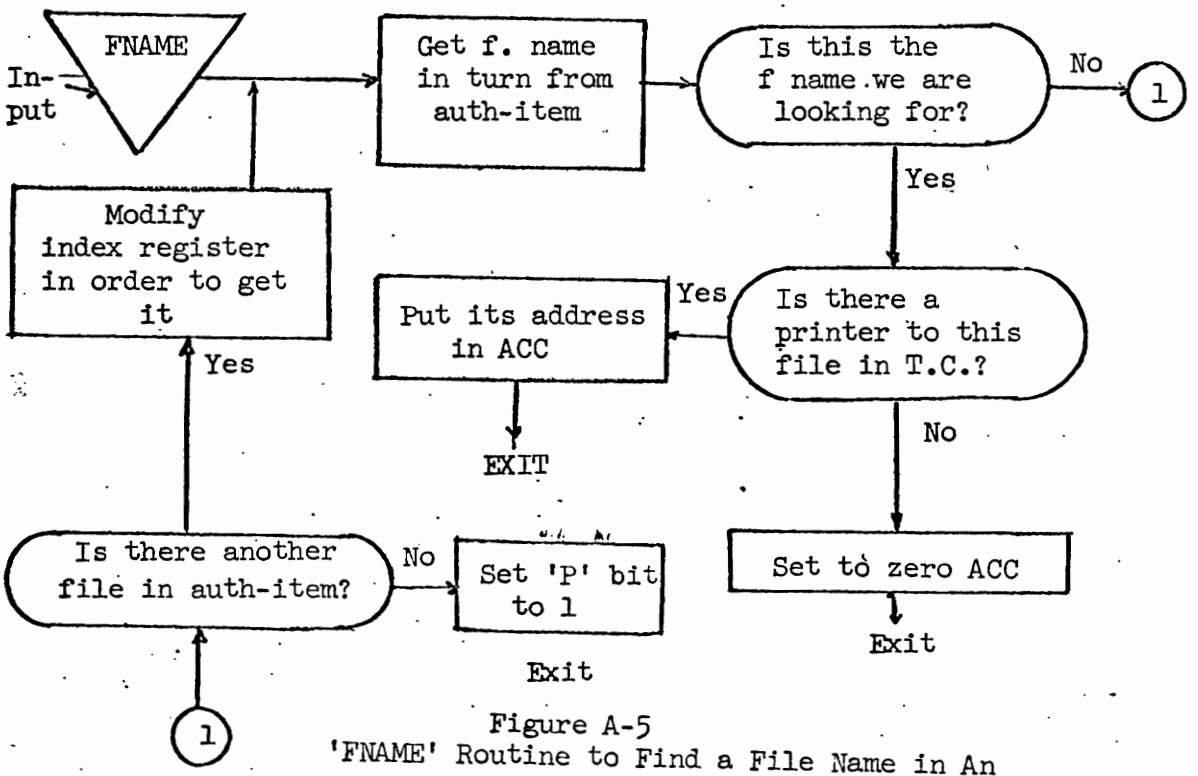


Figure A-5
'FNAME' Routine to Find a File Name in An Authority-Item

Routine that will insert description into a data block corresponding to a given file in corresponding authority-item.

NAME: INSERT

INPUT: 1) Comp. of address of authority item's header in XR1

2) Address of information that will be inserted, in address part of ACC.

3) Size of information to be inserted, in decrement part of ACC.

4) Complement of address of pointer corresponding to data block where the information wants to go.

OUTPUT: 1) This routine will update all the corresponding pointers in auth-item after the insertion has taken place.

2) The comp of address of pointer corresponding to information just inserted will be put in ACC.

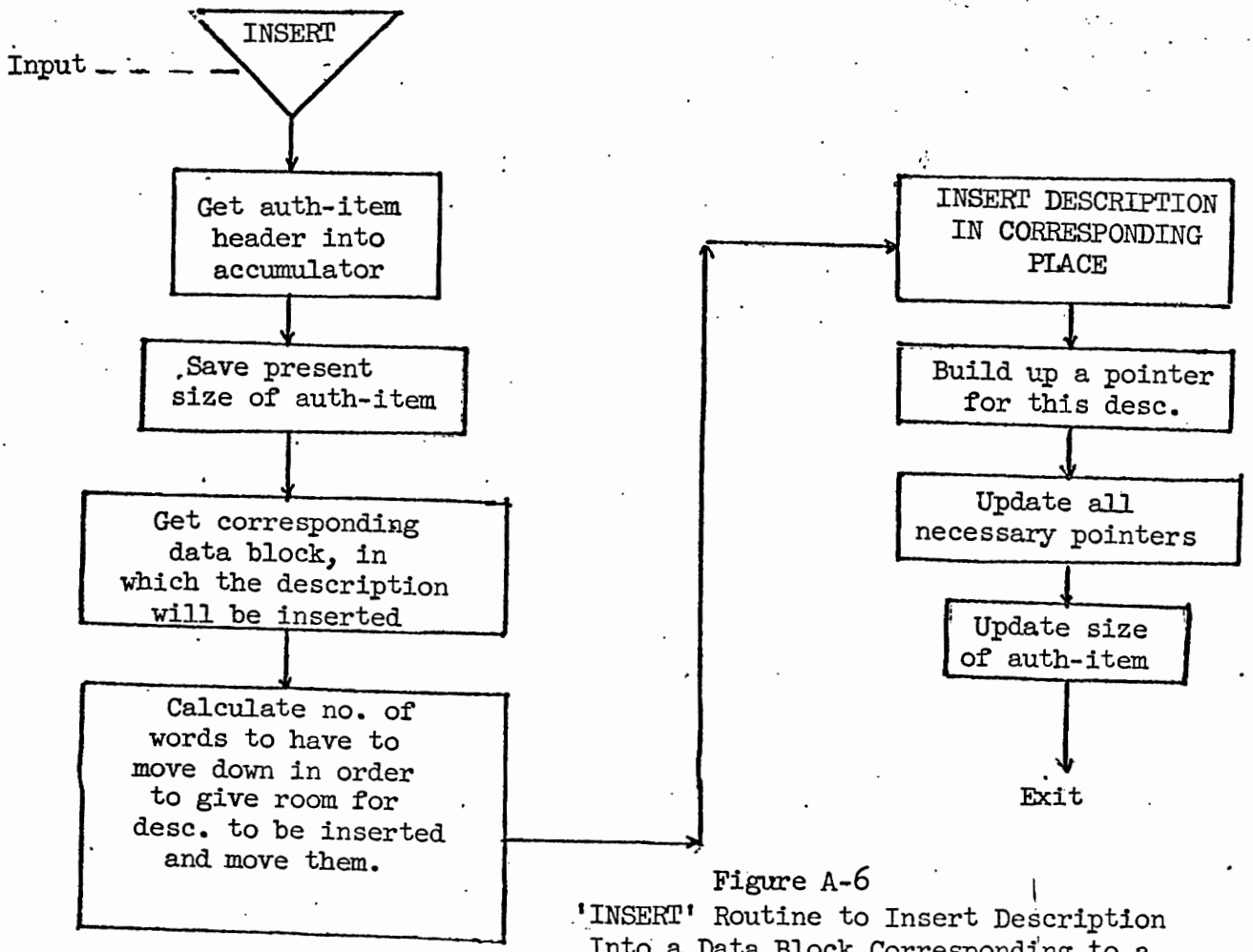


Figure A-6
'INSERT' Routine to Insert Description Into a Data Block Corresponding to a Given File Name in an Authority Item

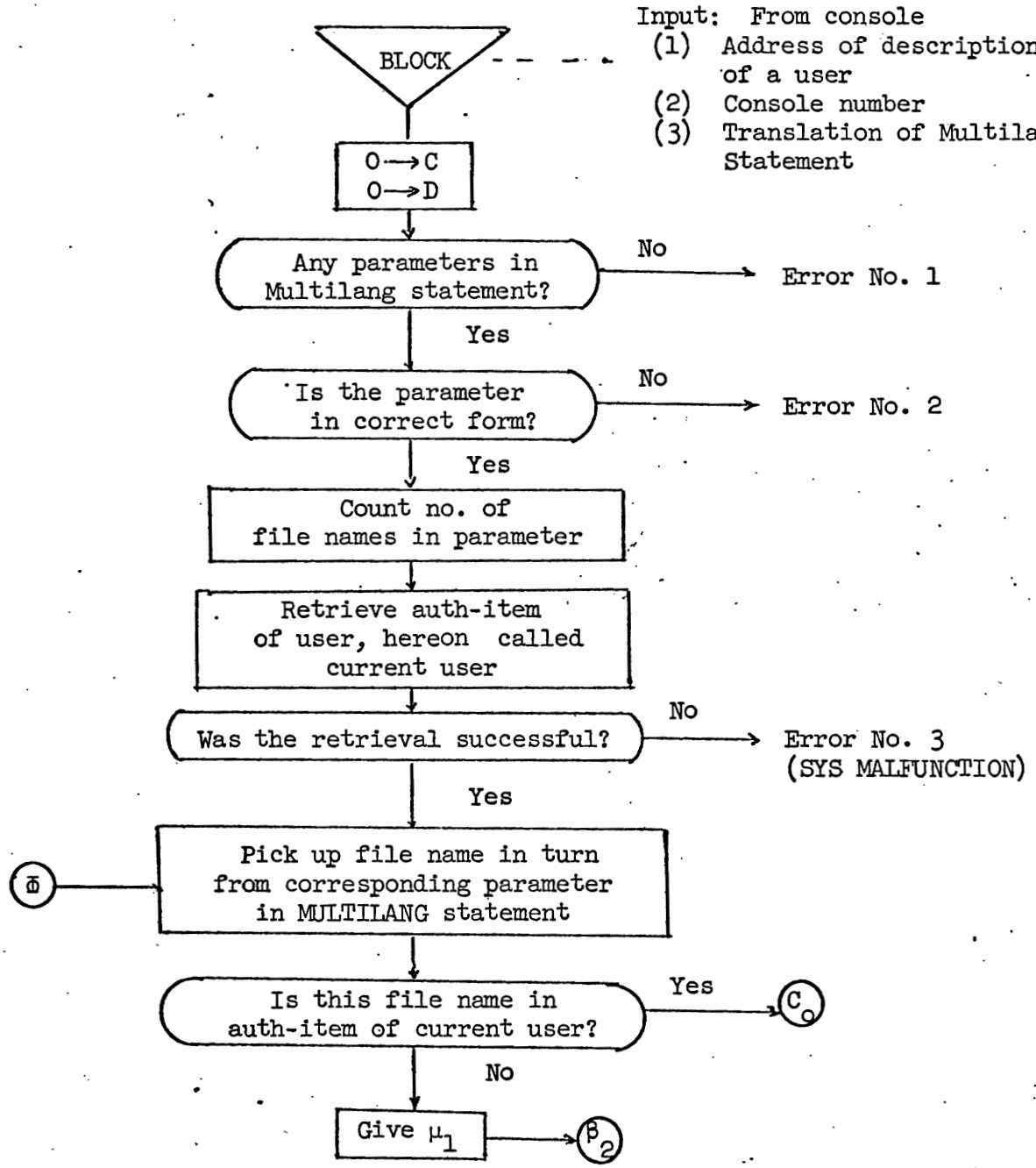


Figure A-7
The BLOCK Program

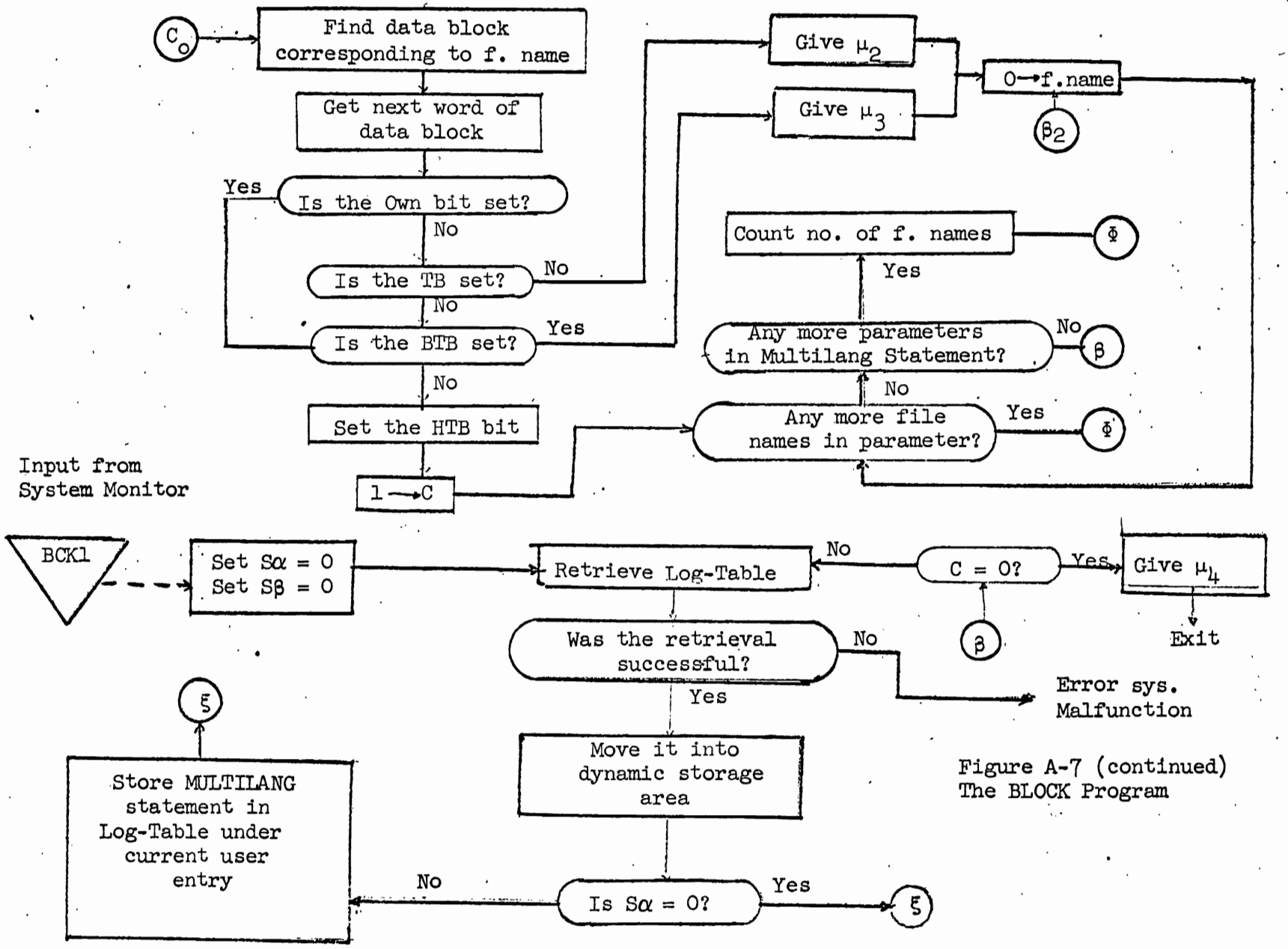


Figure A-7 (continued)
The BLOCK Program

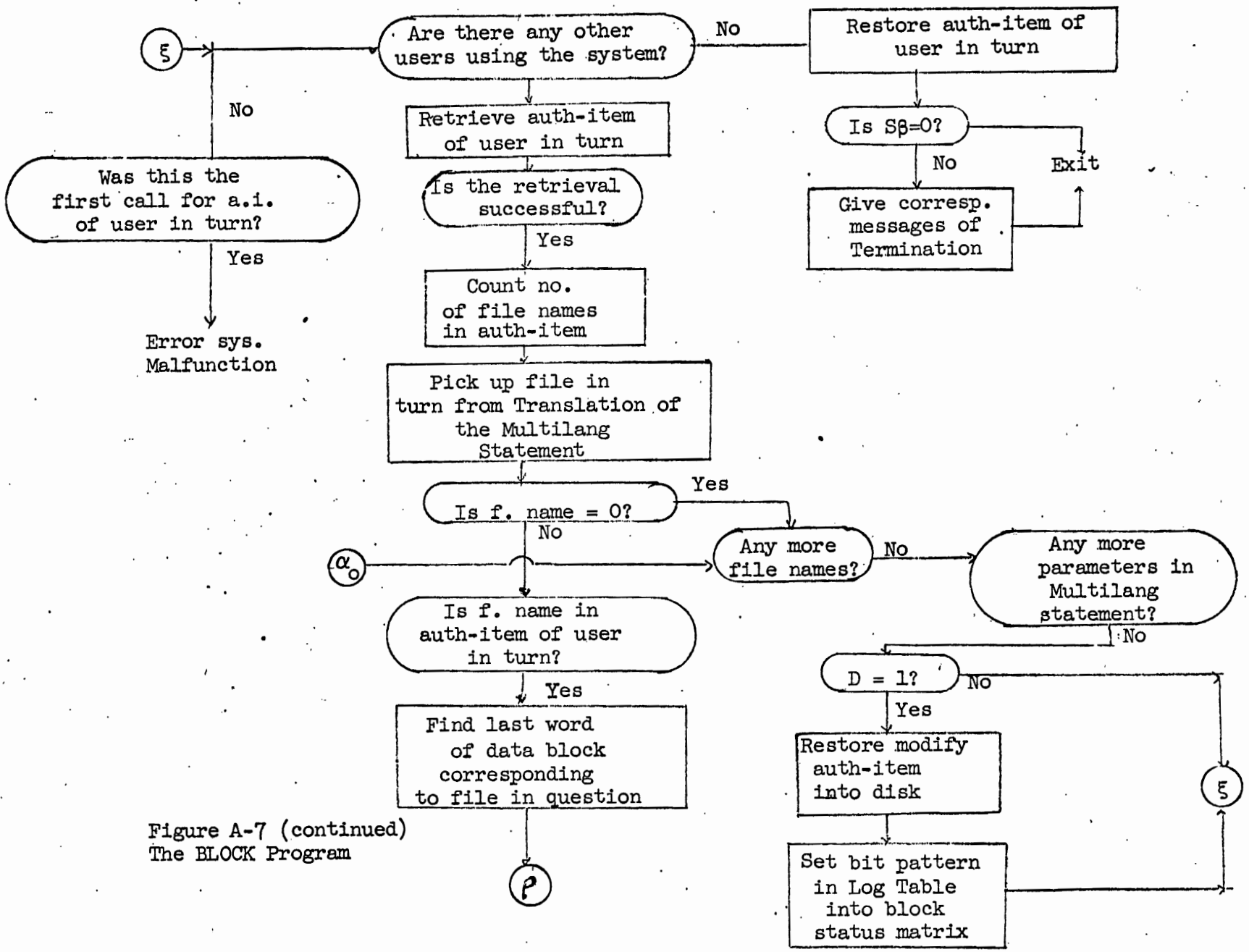


Figure A-7 (continued)
The BLOCK Program

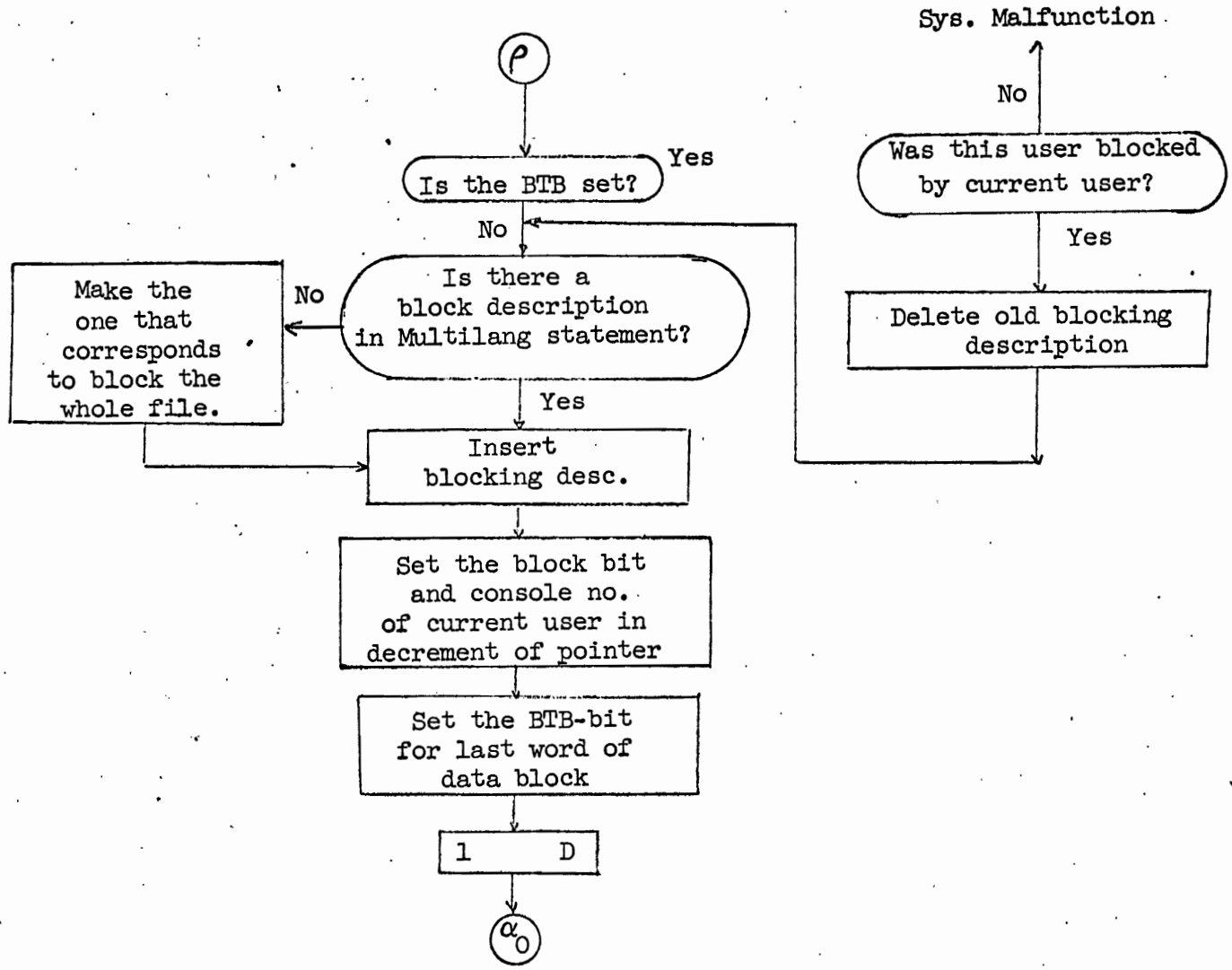


Figure A-7 (continued)
The BLOCK Program

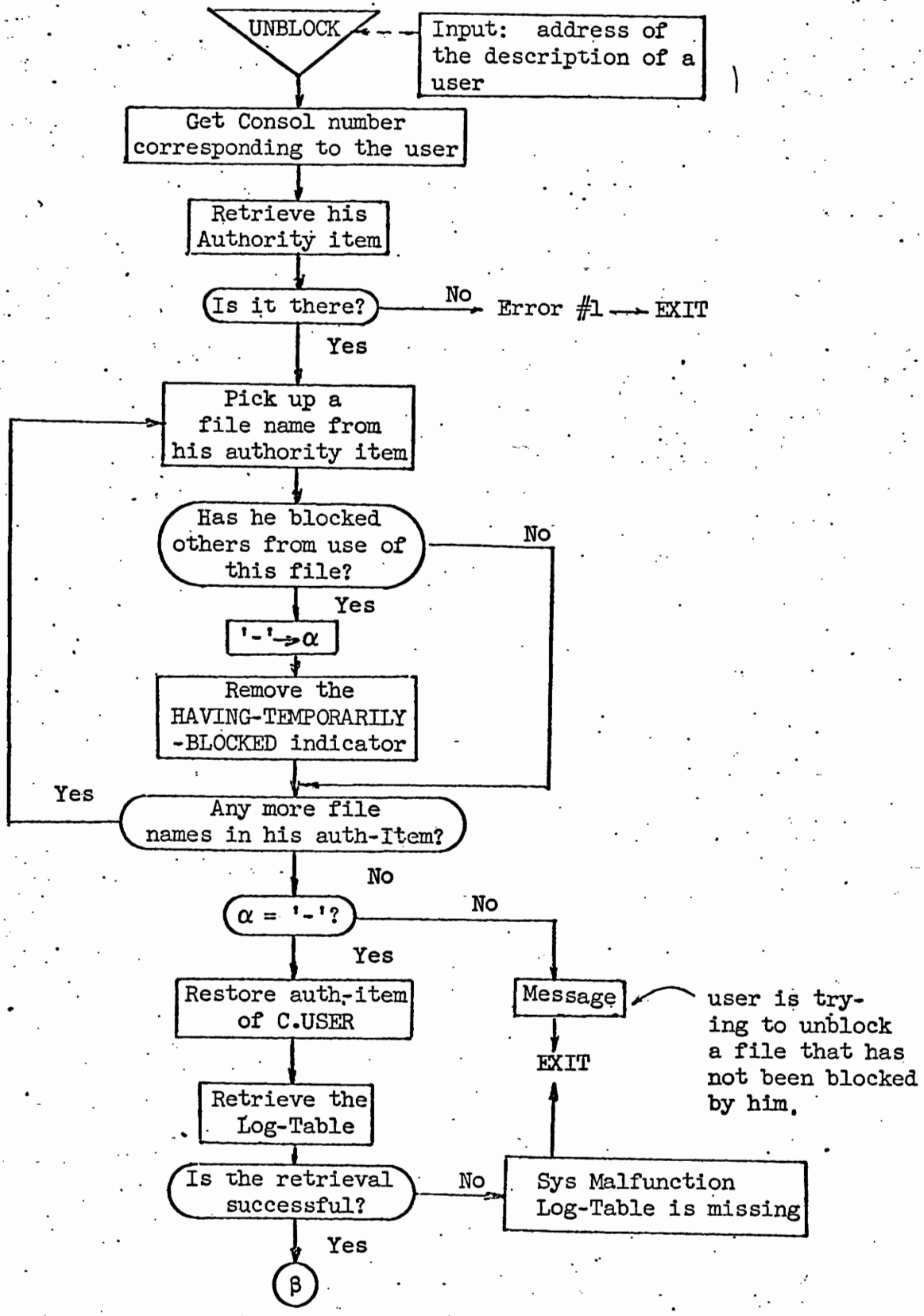


Figure A-8
The UNBLOCK Program

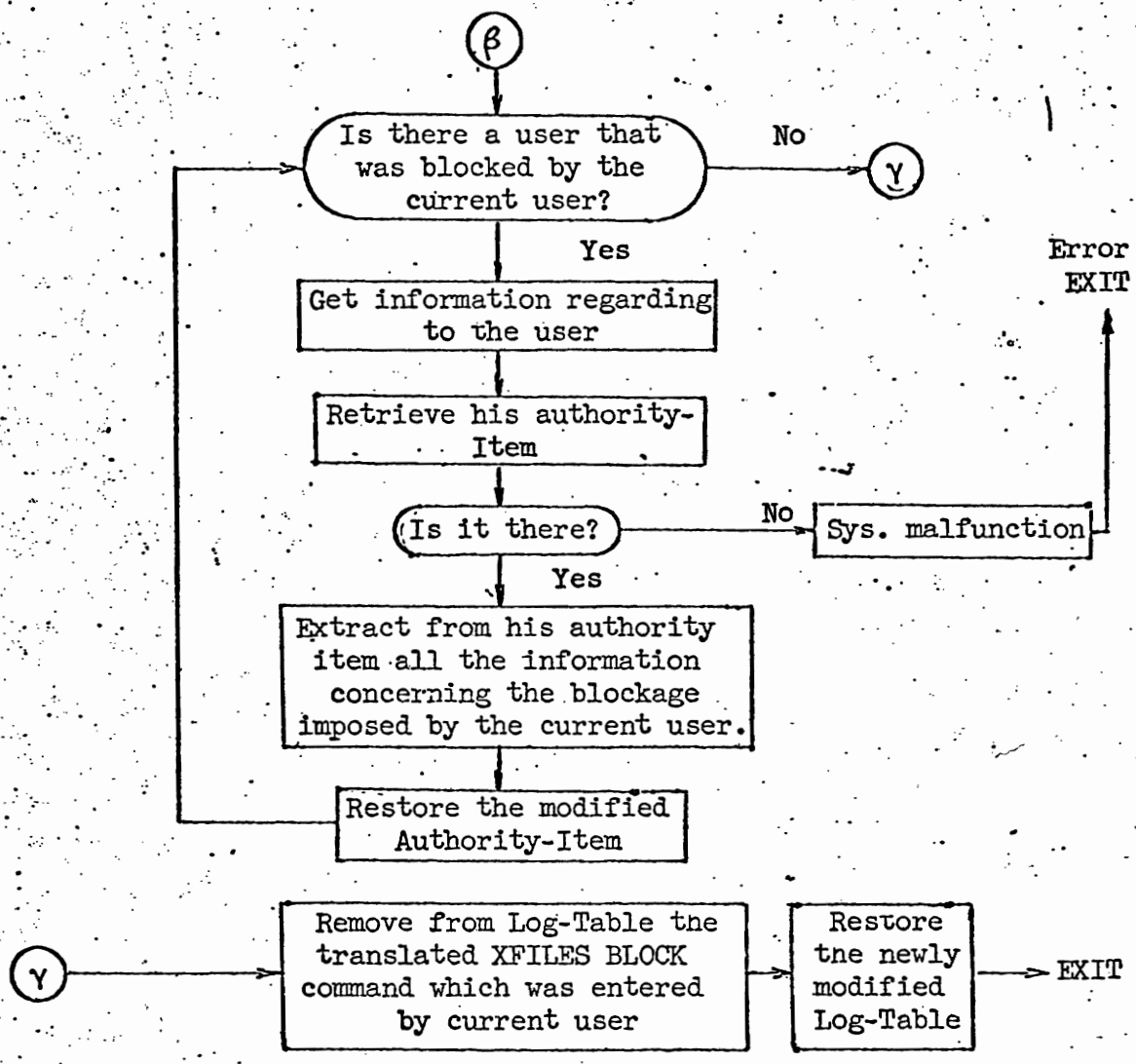


Figure A-8
The UNBLOCK Program (continued)