INFORMATION MANAGEMENT

Information management is concerned with the storage and retrieval of information to the system. The basic functions of information management are

1. Keeping track of all information in the system through various tables, the major one being the file directory called the Volume Table of Contents (VTOC). These tables contain the name, location and accessing rights of all information within in the system.
2. Deciding, policy for determining where and how information is stored and who gets access to the information. Factors influencing this policy are efficientutilization of secondary storage, efficient access, flexibility to users, and protection of access rights to the information.
3. Allocating the information resource.
4. Deallocating of the resource.

FILE SYSTEM:

The modules of information management are referred to as file system. A file system is concerned only with the simple logical organization of information. It deals with collections of unstructured and uninterpreted information at the operating system level.

FILE:

Each separately identified collection of information is called a **file or data set.** File system have evolved from simple mechanisms for keeping track of system with the means of storing, retrieving and sharing information. A file is a collection of related information units **(records).**

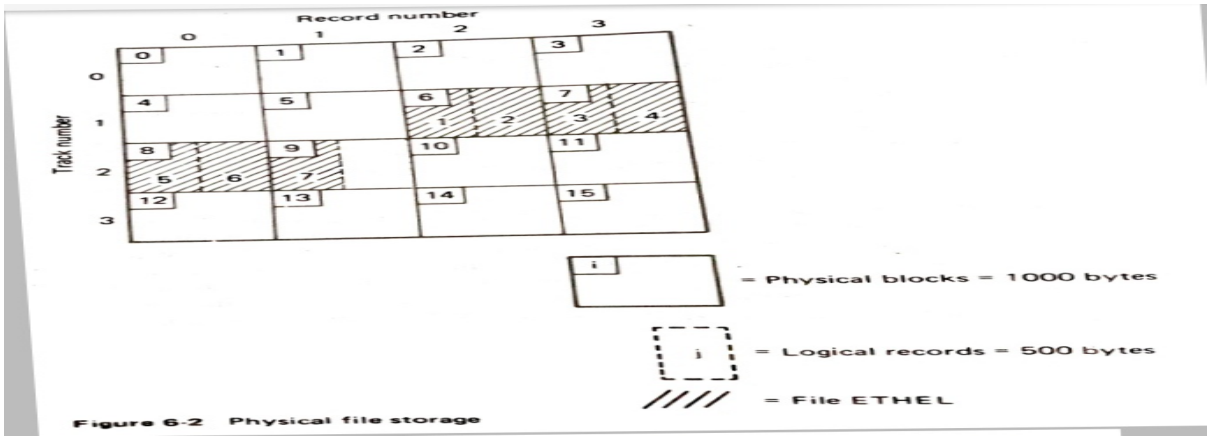


Figure 6-2 Physical file storage

1. GENERAL MODEL OF A SIMPLE FILE SYSTEM:

The general model of a file system is

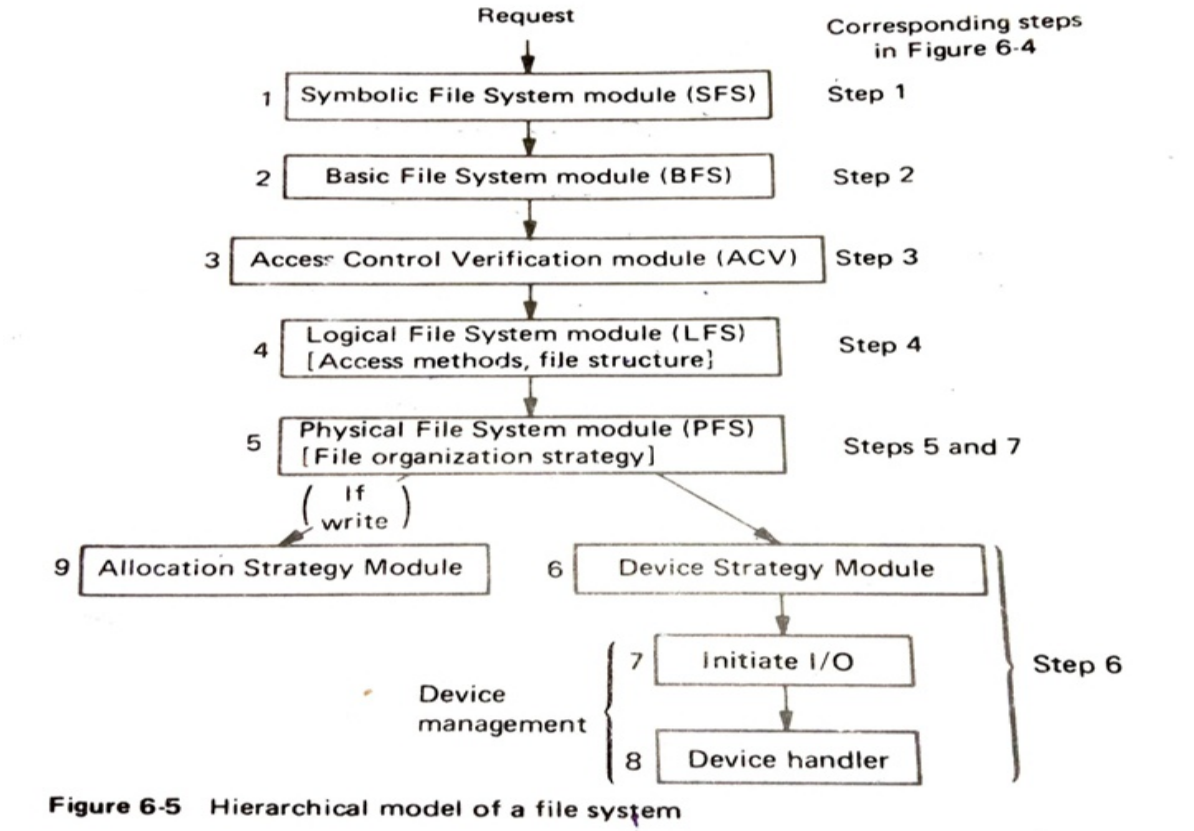


Figure 6-5 Hierarchical model of a file system

a) File Directory Maintenance:

CREATE ETHEL, RECSIZE=500, NUMRECS=7, LOCATION=6

DELETE JOHN.

The CREATE command adds a new entry to the file directory and the DELETE command removes an old one.

b) Symbolic File System:

The first processing step is called the Symbolic File System **(SFS).** A typical call would be:

CALL SFS (function, file name, record number, location),

Such as:

CALL SFS (READ, “ETHEL”, 4, 12000)

The Symbolic File System uses the file name to locate that file's unique file directory entry. It has two separate directories, a **Symbolic File Directory (SFD)** and a **Basic File Directory (BFD).**

Active Name Table (ANT).

	NAME	UNIQUE IDENTIFIER (ID)
1	MARILYN	3
2	ETHEL	5
3	JOHN	6

16 bytes — 4 bytes

20 bytes per entry

(a) Sample Symbolic File Directory

The Basic File System allows operation on files such as:

CALL BFS (READ, 5, 4, 12000) Where 5 is ETHEL'S ID.

ID	LOGICAL RECORD SIZE	NUMBER OF LOGICAL RECORDS	ADDRESS OF FIRST PHYSICAL BLOCK	PROTECTION AND ACCESS CONTROL
1	—	—	0	(Basic File Directory)
2	20	3	1	(Symbolic File Directory)
3	80	10	2	Access by everyone
4	1000	3	3	(Free)
5	500	7	6	Read only
6	100	30	12	Read for MADNICK Read/write for DONOVAN
7	1000	2	10	(Free)
8	1000	1	15	(Free)

(b) Sample Basic File Directory

In summary the symbolic File System is

1. The Symbolic File Directory
2. The Active Name Table
3. Communication with the Basic File System.

c) Basic File System:

The second processing step is called the Basic File System **(BFS).** A typical call would be:

CALL BFS (function, file ID, record number, location),

Such as:

CALL BFS (READ, 5, 4, 12000)

The Basic File System uses the file ID to locate that file's entry in the Basic File Directory (**BFD).** The BFD entries for active files are saved in an Active File Table (AFT).

In summary the symbolic File System is

1. The Basic File Directory
2. The Active File Table
3. Communication with the Access Control Verification module.

d) Access Control Verification:

The third processing step is called Access Control Verification (ACV). A typical call would be

CALL ACV (function, AFT entry, record number, location)

Such as

CALL ACV (READ, 2, 4, 12000)

e) Logical File System:

The fourth processing step is called Logical File System (LFS). A typical call would be

CALL LFS (function, AFT entry, record number, location)

Such as

CALL LFS (READ, 2, 4, 12000)

Logical Byte Address = (Record Number -1) x Record Length

And Logical Byte Length = Record Length.

In summary, the Logical File System is

1. Conversion of record request to byte string request
2. Communication with the Physical File System.

f) Logical File System:

The fifth processing step is called Physical File System (PFS). A typical call would be

CALL PFS (function, AFT entry, byte address, byte length, location)

Such as

CALL PFS (READ, 2, 1500, 500, 12000)

Physical Block Number= Logical Byte address / physical block size +address of first physical block

Physical Block Number=1500 /1000 +6 =1 + 6 =7

In summary, the Physical File System is

1. Conversion of logical byte string request into Physical block Number and Offset.
2. The Physical Organization Table
3. Communication with the Allocation Strategy Module and the Device Strategy Module.

g) Allocation Strategy Module:

The Allocation Strategy Module (ASM) is keeping track of unused blocks on each storage device. A typical call would be:

CALL ASM (POT entry, number of blocks, first block)

Such as:

CALL ASM (6, 4, FIRSTBLOCK)

h) Device Strategy Module:

The Device Strategy Module (DSM) converts the Physical Block Number to the address format needed by the device.

2. SYMBOLIC FILE SYSTEM:

The SFS's function is to map the user's symbolic reference to an ID.

1. Multiple Names Given to the Same File
2. Sharing Files
3. The Same Number Given to Different Files
4. Subdividing Files into Groups

3. BASIC FILE SYSTEM:

When the Basic File System is called, it is given the unique ID of the file requested. The BFS first searches the AFT to determine whether the file is already "opened". If the unique ID assigned to a file is the same as the file's entry number in the Basic File Directory, there is no "searching required to find the entry in the BFD.

4. ACCESS CONTROL VERIFICATION:

Access control and sharing must be considered in conjunction with each other. There are several advantages in allowing a controlled sharing facility:

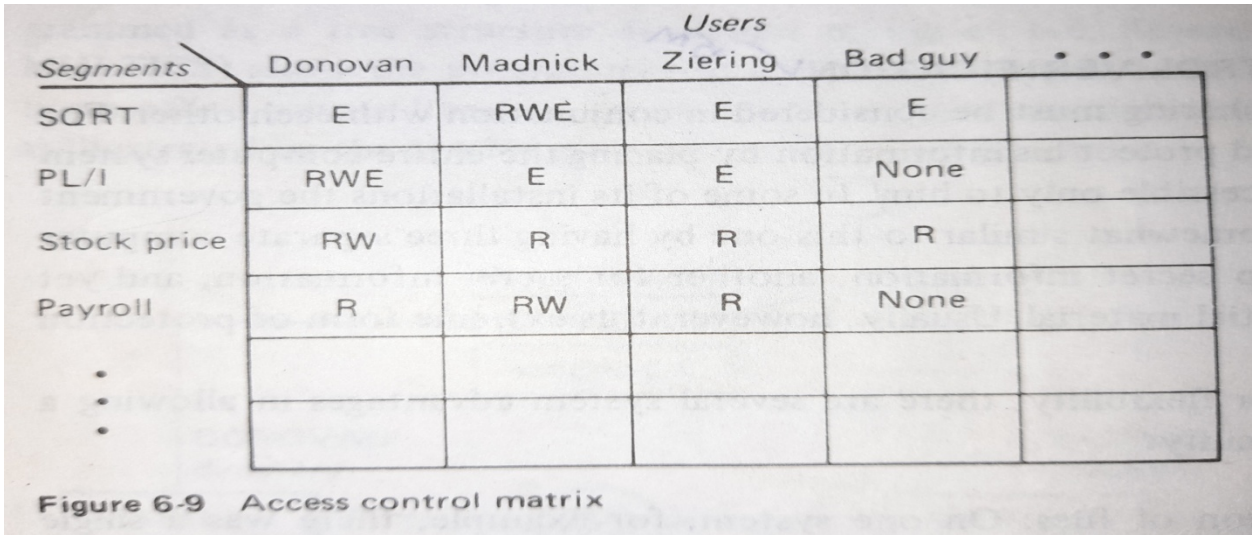
1. It saves duplication of files.
2. It allows synchronization of data operations.
3. It improves the efficiency of system performance. E.g., if information is shared in memory, a possible reduction in **I/O** operations may occur.

A) Access Control Matrix and Access Control Lists:

An access control matrix is a two-dimensional matrix. One dimension lists all users of the computer; the other dimension lists all files in the system. Each entry in the matrix indicates that user's access to that file.

EXAMPLE:

DONOVAN can read file PAYROLL, DONOVAN can both read and write file STOCKPRICE, but MADNICK can read only file STOCKPRICE.



Segments \ Users	Donovan	Madnick	Ziering	Bad guy	. . .
SQRT	E	RWE	E	E	
PL/I	RWE	E	E	None	
Stock price	RW	R	R	R	
Payroll	R	RW	R	None	
. . .					

Figure 6-9 Access control matrix

The Access Control Verification module compares the user's access request with his allowed access to the file- if they do not match, the access is not permitted.

FILE	ACL
DATA	DONOVAN (ALL ACCESS), MADNICK (READ), CARPENTER (READ), ALL OTHERS (NO ACCESS)

The ACL information can be efficiently stored as part of the Basic File directory entry and copied into the Active File Table entry when the file is in use. This makes examination of access control efficient.

B) Passwords:

Access Control Lists, and similar such mechanisms, may take up large amounts of space. An alternative method for enforcing access control is through passwords. Associated with each file in the file directory is a password. The user requesting access to that file must provide the correct password.

C) Cryptography:

Both the passwords and the Access Control List methods are disadvantageous in that the "access keys" are permanently stored in the system. Another method is to cryptographically encode all files. All users may access the encoded files but only an authorized user knows the way to decode the contents.

The decoding (for reading) and encoding (for writing) of the files may be done by the Access Control Verification module. The requesting user provides an argument the code key.

5. LOGICAL FILE SYSTEM:

The Logical File System is concerned with mapping the structure of the logical records onto the linear byte-string view of a file provided by the physical file System.

The access methods are

1. Sequentially structured fixed-length records (sequential and direct access)
2. Sequentially structured variable –length records (sequential and direct access)
3. Sequentially structured keyed records
4. Multiple keyed records
5. Chained structured records
6. Relational or triple-structured records

A) Sequentially structured fixed-length records:

The user views his file as a sequence of fixed-length records. (All records are the same length)

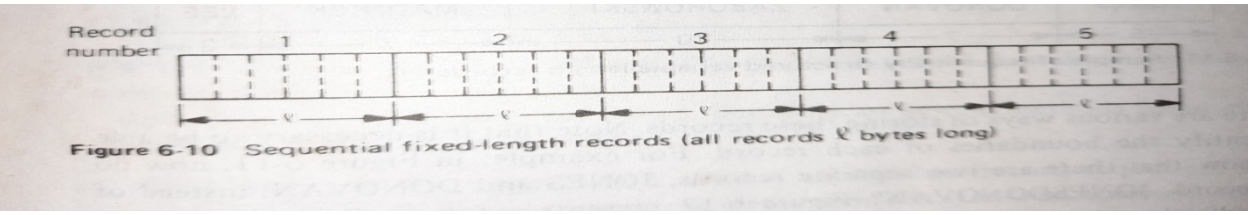


Figure 6-10 Sequential fixed-length records (all records ℓ bytes long)

1) Sequential Access:

For Sequential Access, on each record the user wishes to have the "next" record.

READ FILE (ETHEL) NEXT INTO LOCATION (BUFFER)

The LFS must maintain a Current Logical Byte Address (**CLBA)** in the **AFT** entry for the file.

CLBA = CLBA + RL, where RL is the record length.

2) Direct Access:

For Direct Access the user explicitly specifies the record desired.

READ FILE (ETHEL) RECORD (4) INTO LOCATION (BUFFER)

CLBA = (RN-1) X RL, Where RN is the designated record number.

B) Sequentially structured Variable -length records:

The user views his file as a sequence of variable-length records.

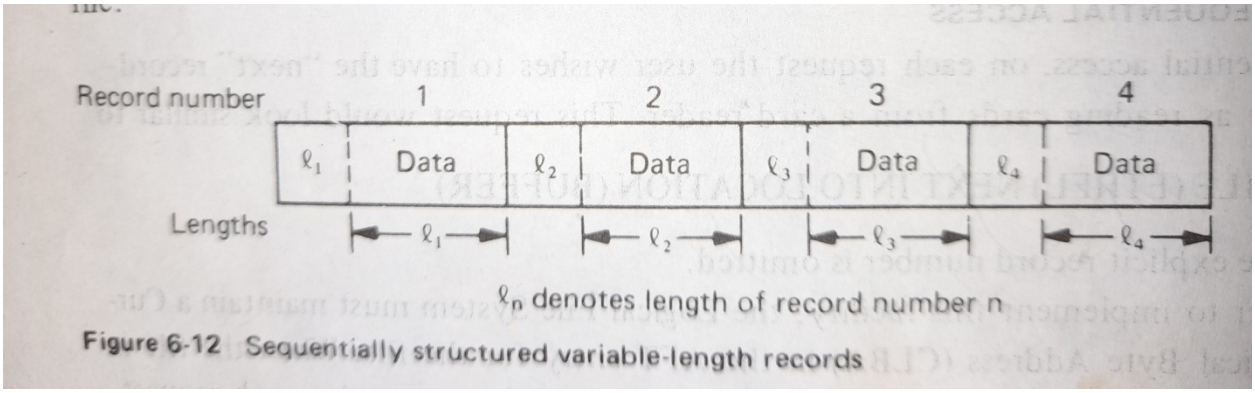


Figure 6-12 Sequentially structured variable-length records

1) Sequential Access:

For Sequential Access,

READ FILE (NAMES) NEXT INTO LOCATION (BUFFER) LENGTH (N)

Where the variable N is set to the length of the record that was read. The length of the BUFFER area must be equal to or larger than the largest record in the file.

CLBA = CLBA + 2 + N

2) Direct Access:

A direct access request would look like

READ FILE (NAMES) RECORD (3) NEXT INTO LOCATION (BUFFER) LENGTH (N)

C) Sequentially structured keyed records:

A different type of access is based on content rather than record number or address.

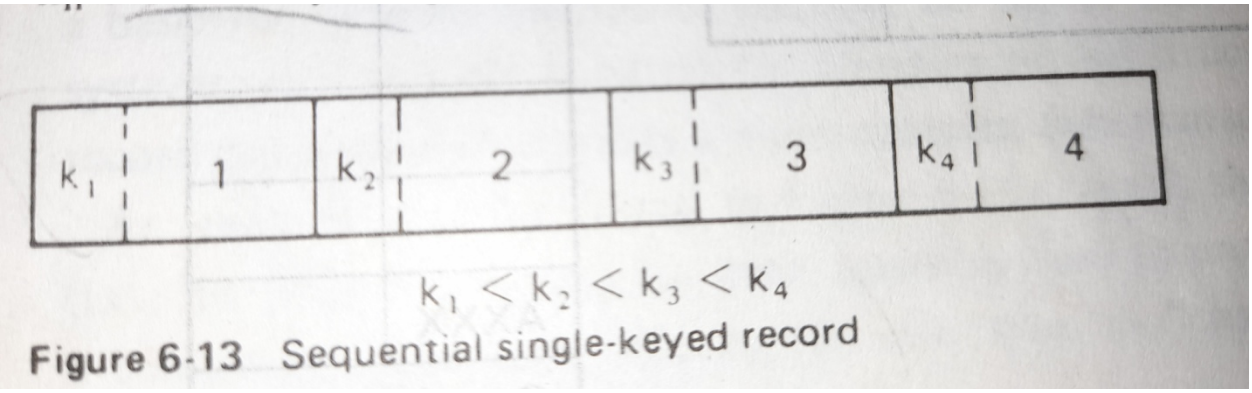


Figure 6-13 Sequential single-keyed record

All records are kept in ascending order based on the keys k1, k2, k3. Etc.,

D) Multiple- keyed records:

The major techniques for

1. Multiple indexes
2. Chaining records with the same keys

E) Chained Structured records:

The keyed and multiple-keyed record structures are not usually offered as part of a basic file system. A chained structured record organization provides a more complex data management facility.

F) Relational or Triple- Structured records:

Another database organization that may be built on top of a basic file system is a relational or triple structure. Logical records may bear a relation to other records.

e.g., Frank is John's father.

RECORD 1	-	RELATION	-	RECORD 2
FRANK	-	FATHER	-	JOHN
JOHN	-	BROTHER	-	PAUL
JOHN	-	FATHER	-	JAMES

5. PHYSICAL FILE SYSTEM:

The Primary function of the PFS is to perform the mapping of Logical Byte Addresses into Physical Block Addresses. The Logical File System calls the Physical File System, passing to it the logical byte address and length of the information requested. The PFS may first call the Allocation Strategy Module (if a write request) and then the Device Strategy Module, or it may call the DSM directly.

The PFS keeps track of the mapping from Logical Byte Address to the blocks of Physical storage.

The three additional considerations are

1. Minimizing I/O operations
2. Allowing logical record size independent of physical block size
3. Allowing noncontiguous allocation of file space.

A) Minimizing I/O operations:

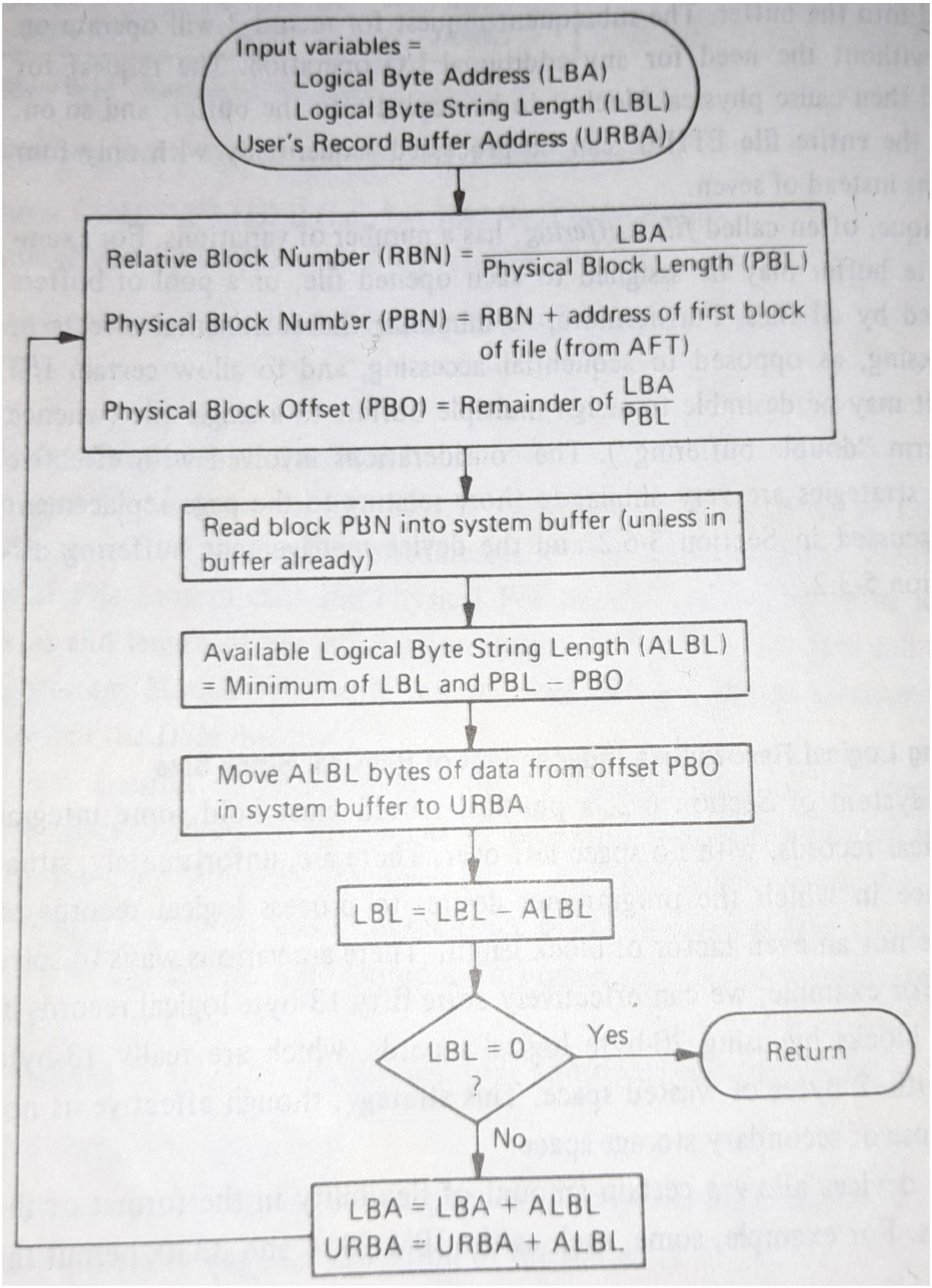
If all the records of file ETHEL were to be read sequentially, it would require seven I/O operations (each operation reading one physical block) to read the seven logical records using the algorithm.

Since the entire file occupies only four physical blocks, we could reduce the number of I/O operations by eliminating redundant operations.

The number of I/O operations can be reduced by keeping track of which physical blocks are in core. For each I/O read operation a physical block is copied into a buffer in main memory. This technique, often called file buffering, has a number of variations.

Example:

A separate buffer may be assigned to each opened file, or a pool of buffers may be shared by all files.


Input variables =
Logical Byte Address (LBA)
Logical Byte String Length (LBL)
User's Record Buffer Address (URBA)
Relative Block Number (RBN) = LBA / Physical Block Length (PBL)
Physical Block Number (PBN) = RBN + address of first block of file (from AFT)
Physical Block Offset (PBO) = Remainder of LBA / PBL
Read block PBN into system buffer (unless in buffer already)
Available Logical Byte String Length (ALBL) = Minimum of LBL and PBL – PBO
Move ALBL bytes of data from offset PBO in system buffer to URBA
LBL = LBL – ALBL
LBL = 0 ?
Yes
Return
No
LBA = LBA + ALBL
URBA = URBA + ALBL

B) Allowing logical record size independent of physical block size:

A physical record must hold some integral number of logical records, with no space left over. Many storage devices allow a certain amount of flexibility in the format of the individual tracks.

This approach also has the benefit of allowing logical records that are larger than the maximum size physical block.

C) Allowing noncontiguous allocation of file space:

Contiguous allocation of file space is not always possible. If the physical blocks of a file are not contiguous, a different algorithm is needed to map from logical byte address to physical block number. Two popular techniques are

1) Chained blocks and
2) File maps.

A) CHAINED BLOCKS:

In a chained blocks mapping, each physical block contains the address of the next "logically contiguous" block. The Basic File Directory entries contain the address of the first physical blocks, as before, but subsequent block addresses are found by means of the address pointers (often called chains).

The address pointer may be stored within the physical block, thereby decreasing the data capacity of the block slightly (holds only 998 bytes per block instead of 1000 bytes), or as part of the header field (key field) of the data block which is possible on many devices.

The chained blocks approach is efficient for sequential access to the blocks but quite inefficient for direct access for reasons similar.

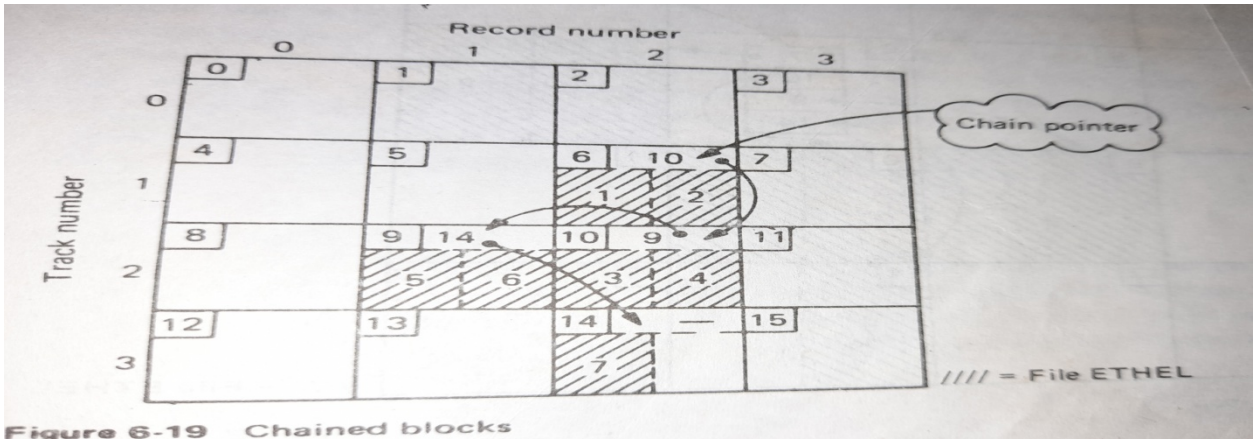


Figure 6-19 Chained blocks

B) FILE MAPS:

Another approach to noncontiguous allocation is to use a File Map Table to map each Logical Block Address to its physical block address. This file map may be stored as part of the entry in the BFD or in a separate block.

When the file is opened, all or part (if the file is especially large) of the file map is copied into memory as part of the Active File Table information.

Physical Block Address = File Map (Logical Byte Address/ block size)

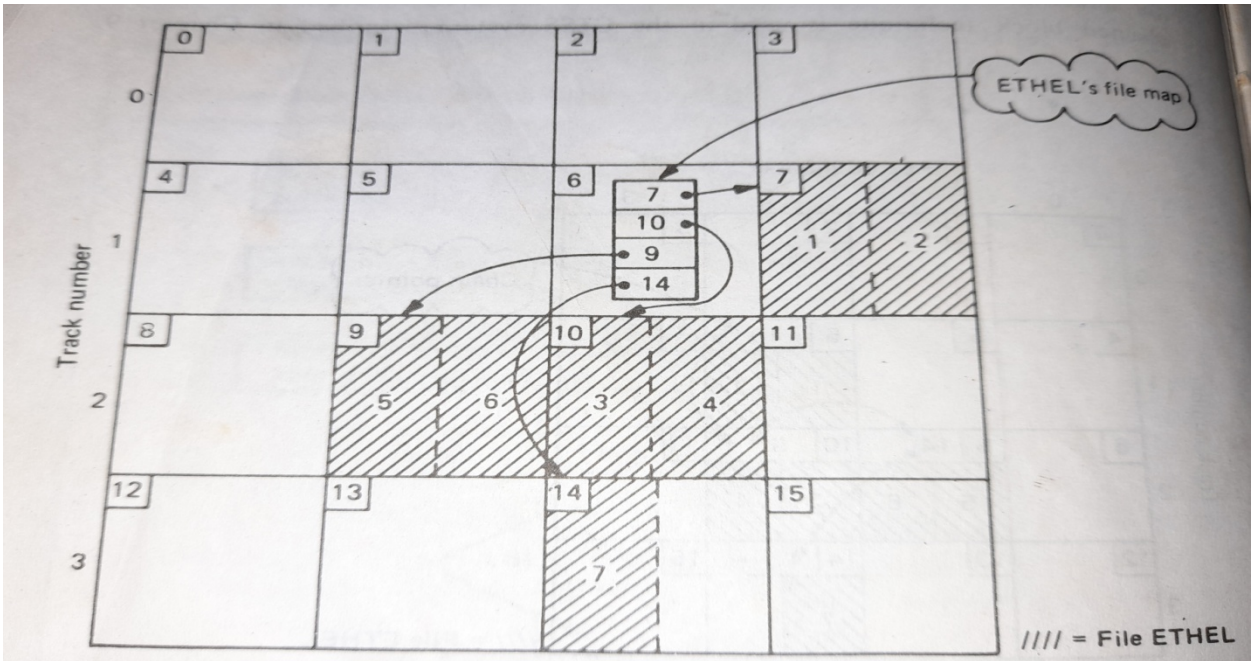


Figure 6-20 *File map*

6. DEVICE STRATEGY MODULE, I/O INITIATOR, DEVICE HANDLER:

The basic functions of these routines are as follows:

1. Map the physical address into a device address.
2. Create the channel program

   e.g., a program to do the following:

   a. seek to track 1

   b. Search track 1 for record 3

   c. wait until disk reaches it

   d. read

3. Request the I/O, that is, all the I/O scheduler to schedule this I/O.
4. Handle the I/O interrupts and error conditions that may occur.