DOCUMENT RESUME

ED 026 868 64 EM 007 093

By-Sullivan, Todd

A Prototype System for a Computer-Based Statewide Film Library Network: A Model for Operation. Statewide Film Library Network: System-1 Specifications - Files.

Syracuse Univ., N.Y. Center for Instructional Communications.
Spons Agency-Office of Education (DHEW), Washington, D.C. Bureau of Research.
Report No-SD-003-0
Bureau No-BR-7-0259
Pub Date 30 Jun 68
Contract-OEC-1-7-070259-2656
Note-58p.
EDRS Price MF-$0.50 HC-$3.00
Descriptors-*Computer Oriented Programs, Computer Storage Devices, *Data Processing, *Films, Information Networks, *Information Storage, Input Output, Interlibrary Loans, Library Equipment, Library Facilities, Library Reference Services, Library Science, *Library Technical Processes, Magnetic Tapes, Systems Analysis
Identifiers-IBM System 360 Model 50 Computer, *Statewide Film Library Network, SUFRL, Syracuse University Film Rental Library, Teletype Model 33 ASR Teletypewriter

Using an IBM System/360 Model 50 computer, the New York Statewide Film Library Network schedules film use, reports on materials handling and statistics, and provides for interlibrary loan of films. Communications between the film libraries and the computer are maintained by Teletype model 33 ASR Teletypewriter terminals operating on TWX (Teletypewriter Exchange) service, programed mainly in S/360 Assembler Language. This configuration permits on-line (real time) tele-processing, the central concept in the system. The files (S/360 data sets) are stored on either direct access storage devices (disks) or sequential access storage devices (magnetic tape) according to their use in processing. Within the Network's system, there are three classes of files: system files, containing system programs and backup files; input/output files, containing tables for I/O editing and temporary I/O storage; and the data files, containing information about films, libraries, customers, and bookings. Appendices: list of files by sort sequence, notes on System/360 organization and data storage, and notes on disk storage devices. (TI)

STATEWIDE FILM LIBRARY NETWORK:

System-1 Specifications - Files

Todd Sullivan

CENTER FOR INSTRUCTIONAL COMMUNICATIONS · SYRACUSE UNIVERSITY

ED0 26868

STATEWIDE FILM LIBRARY NETWORK:

System-1 Specifications - Files

Todd Sullivan

EM007093

A Prototype System for a Computer-Based Statewide Film Library Network: A Model for Operation

Document Number SD-003-0

June 30, 1968

U. S. Office of Education
Research Project at the
Syracuse University
Center for Instructional Communications
121 College Place
Syracuse, New York 13210
315/476-5541 ext. 3807

This research project is supported by the U. S. Office of Education under Grant number OEC-1-7-070259-2656.

This document was compiled by Todd Sullivan under the guidance of Dominick W. Auricchio, and with the assistance of Muriel L. Day, James P. Eadie, David B. Greene and Peter B. Olson.

CONTENTS

1. INTRODUCTION

The Statewide Film Library Network is being developed in New York State to provide better service to school teachers using 16 mm educational films. The Network is designed to include film libraries at three levels: local libraries located in Boards of Cooperative Educational Services (BOCES) throughout New York State, regional libraries at large public libraries and State University of New York campuses, and a statewide library at Syracuse (the Syracuse University Film Rental Library - SUFRL).[1]

Services provided through the Network's computerized system include film use scheduling, materials handling reporting, extensive statistical reporting on film usage, and the inter-library loan of films to backstop requests for film usage which would otherwise go unfulfilled. (The system will be expanded eventually to include bibliographic and catalog production services, as well as handling other forms of educational media.)

The computerized system is currently being operated at the Syracuse University Computing Center, using an IBM System/360 Model 50 computer which has the following configuration:

(a) one 2050 Processing Unit which provides 262K bytes (actual) of main storage, divided into four variable-sized partitions under the MFT-I (release 14.0) version of the OS/360 operating system; (the Network's system resides in an upper partition of 43,008 bytes);
(b) one 2314 Direct Access Storage Facility;
(c) six 2402 Magnetic Tape Units (four 9-channel and two 7-channel units);
(d) one 2540 Card Read Punch;
(e) one 1403-N1 Printer;
(f) various remote I/O devices linked to the computer by one 2848 Display Control Unit and one 2701 Data Adapter Unit.

---

1 At the time this document was issued, four libraries were participating in the Network: Erie County BOCES No. 1 at Buffalo, Suffolk County BOCES No. 3 at Huntington, Westchester County BOCES No. 1 at Yorktown Heights, and SUFRL at Syracuse.

Communications between the film libraries and the computer are maintained by Teletype model 33 ASR Teletypewriter terminals operating on TWX (Teletypewriter Exchange) service. The programming of the system has been done predominantly in S/360 Assembler Language, with a few programs written in PL/I.

This configuration permits on-line (real-time) teleprocessing which is the central concept in the system. The real-time or telecommunications mode of operation provides the user with immediate service via the Teletype terminal, wherein the system makes the results of processing available to the user within seconds after the receipt of input. It is planned that the system will also operate in another mode - the batched mode - where input and output occur at the computer site. This mode will provide non-immediate service for high volumes of data at a lower cost, as the expense of telecommunications is eliminated.

2. SYSTEM DATA STORAGE REQUIREMENTS

The Statewide Film Library Network is based on a real-time system of computer programs which operate on input submitted to the system, the output generated from the input, and various kinds of stored data about films, the Network libraries, and their customers. These four elements of the system - programs, input, output, and stored data - comprise the subject matter of the information that is stored in the files associated with the system.

Since the system normally operates on a real-time basis in a relatively small partition of main storage in the S/360 computer, and since the total main storage required for all the system's programs far exceeds the 43,008 bytes of storage available in the partition, the system's programs are broken into a large number of modules which are stored on disk and usually are called into the partition only as they are required. Thus, the system's programs and associated information comprise one category of files associated with the system.

The input and output which the system user sees is substantially different in form from what the S/360 computer and the Network's system require for internal manipulations. There exists, then, a set of programs which perform transformations (editing) on all input submitted to the system and all output emitted by the system. In order that the system's input and output be flexible with respect to format and content, these editing programs are generalized in that they are interpretive; that is, these programs use computer-stored tables describing the various input and output formats in performing the editing functions. Also, since certain operations performed by the system's programs involve information stored in the data files, tables exist which similarly describe the formats of the various data files. These descriptive tables for input, output, and data file formats comprise another category of files associated with the system.

Finally, since the system's largest class of operations involve the scheduling of requests for film usage, there is a third category of files which contain data pertinent to the scheduling operations - information about the films, bookings made on the films, the film libraries, and the libraries' customers. These files are called the data files.

Prior to the discussion of the exact contents, formats and usages of the files associated with the system, it will be appropriate to investigate the general organization of all the files in the system, and the various means of storing and accessing the information contained in them.

3. FILE ORGANIZATION

The files, or data sets in S/360 terminology, which are utilized by the system are stored on either direct access storage devices (in this case, disk) or sequential access storage devices (magnetic tape), according to their use in processing. Again, the system incorporates the capability for real-time teleprocessing, and any files which may be referenced during real-time operations are stored on disk in order to ensure relatively fast access to them. All other files may be stored on magnetic tape.

The files in the system may also be divided into two classes according to their method of logical access: direct-access and sequential-access. All direct-access files are stored on disk, and all are used during real-time operations. They are generated and accessed using a very basic level of data access provided by OS/360,[2] called Execute Channel Program (EXCP). Some of the sequential-access files are stored on disk, and are used during real-time operations; others are stored on magnetic tape, and are used only during batched processing operations. All sequential-access files are generated and accessed using either the Basic Sequential Access Method (BSAM) or the Queued Sequential Access Method (QSAM), as provided by OS/360.[3] The Basic Telecommunications Access Method (BTAM) is used in the system to handle the I/O interface between the telecommunications data transmission devices and the programs of the Network's system.

In order to provide a relatively simple and uniform method of data access for all files in the Network's system, a package of higher level I/O routines has been developed. Basically, the package utilizes BSAM, QSAM, BTAM, and the system's own access method for direct-access files. The package handles all program calls for input and output by providing the programmer with a set of

---

2 OS/360 is IBM's designation for the System/360 Operating System.

3 The key to this whole proposition lies in the fact that files stored on direct access storage devices can be accessed in either direct or sequential fashion, whereas files stored on sequential access storage devices can be accessed only in sequential fashion.

macro-instructions for reading and writing data. While a detailed discussion of the I/O package is outside the scope of this document, the remainder of this section is devoted to a relatively complete account of the logical and physical organization of the system's direct-access and sequential-access files.

(It is suggested that Appendix B be reviewed at this point if the reader is unfamiliar with the organization of main storage in the S/360 computer.)

3.1 Direct-Access Files

LOGICAL ORGANIZATION

The logical organization of any given direct-access file is determined by the various attributes of the records contained in it. In the Network's system, there are three attributes of logical records: type, format, and length.

Conceptually, any direct-access file may be considered as a single block of data which is divided into some finite number of logical records. These records will be of either one or two types, depending on the data contained in the file. Those files which contain records of two distinct types are said to be composed of header records and detail records. In such a file, these records are grouped in a hierarchy, with headers being the major elements and details the minor elements, where all the details belonging to a header follow immediately after that header in the file (see Figure 1).

HEADER 1	Detail 1.1	Detail 1.2	Detail 1.3	HEADER 2	Detail 2.1	Detail 2.2	Detail 2.3	Detail 2.4

Figure 1. Logical format of a direct-access file showing two header records and their respective detail records.

Within any given file, both types of logical records must conform to one of two formats. If the records (either headers or details) are of uniform length throughout the file, they are specified as beings of the fixed length format, whereas if they are of various lengths, they are of the variable length format.

In addition to specifying the format attribute, it is necessary to indicate the actual length in bytes of every record in each file. Since records of the fixed length format are uniform in size, their length may be specified generally for all such records in a file. (If both the headers and details in the same file are of the fixed length format, their lengths must be specified independently.)

On the other hand, it is necessary to indicate the size of each individual record which is of the variable length format, as such records are not uniform in size. Here, the record length is contained in the first two bytes of data in the record proper.

Logical records are identified by means of a record ID which has two elements: a header ID and a detail ID. A header ID is equivalent to the header record key which is a data element not exceeding one fullword in length, the value of which is equal to some integer greater than 1 and less than $(2^{31}-1)$. A detail ID is composed of one or two detail record keys,[4] each of which is a data element not exceeding one halfword in length and equal in value to some integer greater than 1 and less than 2^{16}. Figure 2 depicts the contents and physical organization of a record ID.

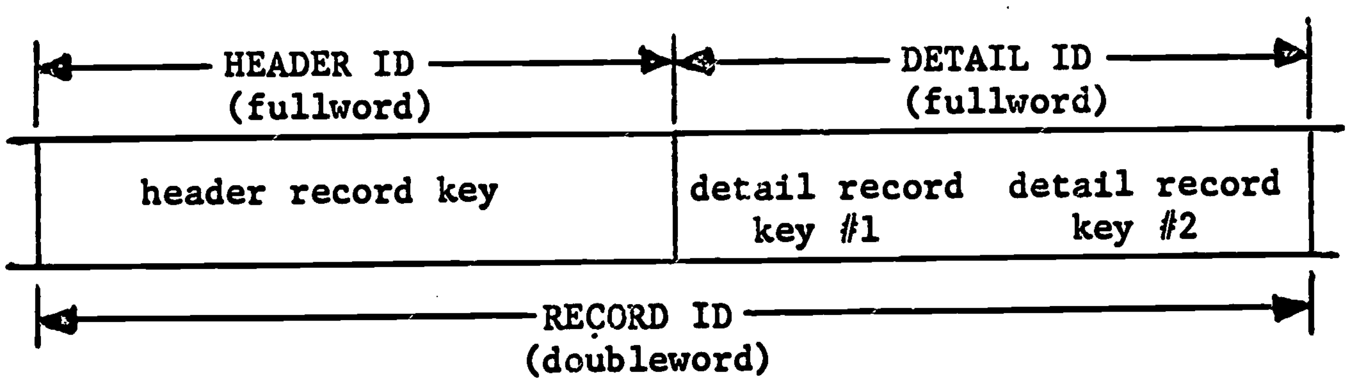

Figure 2. Record ID, showing physical and logical organization.

4 Some detail records require only one key (data element) for identification, while others (e.g., detail records in the Short Catalog & Bookings File) require two keys for complete identification.

A header record, then, is identified by a record ID where the header ID is equal to the header record key and the detail ID is equal to zero. A detail record is identified by a record ID where the header ID is equal to the key of the header record to which it belongs in the hierarchical structure of the file, and the detail ID is equal to the detail record key(s). Note that the header/detail ID's must be in ascending algebraic order, and no two records may have the same ID.

In order to facilitate operations on a file, two dummy records are provided. The first record, located at the beginning of the file, has header ID = 0, detail ID = 1, and the second, at the end of the file, has header ID = $(2^{31}-1,)$ detail ID = 0. A file which contains only these two records is considered to be empty.

Note that these records may not be used to store data.

When the programmer wishes to access data in a file, he has two options. He may explicitly specify the header and detail ID's of the record(s) he desires, or, once having located a record, he may retrieve succeeding records in sequence with automatic feedback of the ID's. The first mode of access is called the random-access mode, and the second is the sequential mode. Once a record has been located and loaded into main storage, the programmer may modify its content and rewrite it back into its original place in the file. Furthermore, if the programmer desires to create a new record, he may do so, and insert it in logical sequence into the file, provided that it does not already exist (i.e., the record must have a unique ID).

PHYSICAL ORGANIZATION AND DATA ACCESS[5]

An entire file is split into physical records (or blocks) of uniform length, in this case, 256 bytes. Logical records are packed contiguously into blocks.[6] In the interests of efficiency, some slack (a maximum of 4 bytes per block) may be left at the end of any given block.

A certain amount of control information is needed to locate and identify logical records within a block. Part of this control information is stored in the first ten bytes of each block; the remainder is stored with <u>each</u> logical record in a halfword prefixed to the logical record. The first two bytes of the ten-byte control section contain pointers to the first byte of the first complete logical record in the block, and the last byte of the last complete logical record. The remaining eight bytes contain two fullword bases, one for the header ID's and one for the detail ID's, which together contain the record ID of the first logical record in the block (see Figure 3).

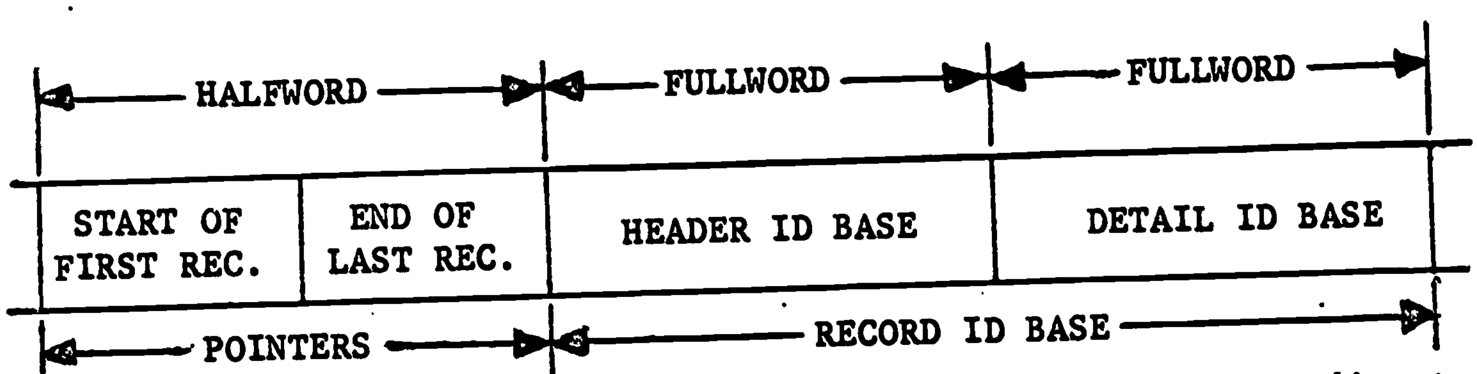

Figure 3. Ten byte control section of a physical record in a direct-access file.

---

5 The following discussion involves certain attributes of disk storage devices; if the reader is unfamiliar with disk units, he is encouraged to read Appendix C before progressing further.

6 This method of physical organization facilitates the dynamic buffering which is performed upon the files. Also, as a result of this packing scheme, logical records of any practicable length (greater than 256 bytes) may be accomodated in the files.

The halfword of control information prefixed to each logical record contains three elements: (a) a one-bit flag identifying the record as a header or detail record, (b) a one-bit flag which may indicate that an addition to the file has been made immediately following this record, and (c) a 14-bit element containing an increment used to calculate the true ID of the record.

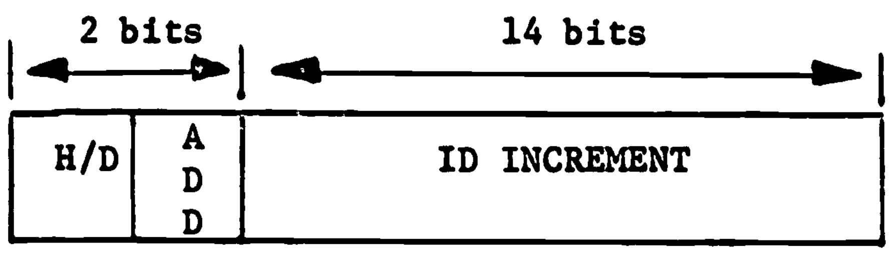

Figure 4. Halfword prefix of control information.

Access to any given logical record in a block is performed by routines which scan the block and add the 14-bit increment from each logical record to the appropriate fullword base.[7] (As each new header record is hit, the increment is added to the header base, and the detail base is reset to zero.) Scanning proceeds until either a match is found, or a record having an ID greater than that of the record sought is found. The latter condition is denoted "record not found".

Note that if the ID's of any two consecutive logical records vary more than $(2^{14}-1)$, this scheme breaks down, as the ID increment is too large to fit in the increment field. To overcome this problem, if the ID's of two logically consecutive records do vary more than $(2^{14}-1)$, the first of the two records is placed in the current physical block, the remainder of the block is filled with slack zeroes, and the second record begins a new block.

7 These 14-bit increments are absolute (i.e., non-cumulative) for each logical record. That is, for example, if the scan routine were searching for a logical record n, with a Key of K_n, in a block with a base B, then $K_n = B + I_n$, where I_n is the increment prefixed to logical record n. (If the increments were cumulative, $K_n = B + I_0 + I_1 + \ldots + I_n$ would be the case, where I_i is the increment prefixed to logical record i.)

The first block on each disk track contains an additional section of control information which is suffixed to the standard ten-byte control section. This additional section is called the track index, and consists of additional header/detail bases for each block on the track. The convenience of the track index lies in the fact that, once a record is known to be on a given track, the record may be located in at most two disk reads - one for the track index, and one for the block containing the record sought. (If the track index did not exist, a sequential search of all the blocks on the track would be necessary.)

It remains to find which track contains the record sought. The solution lies in the track table which effectively contains the header ID's for the first logical record on each disk track on which the file is stored. The track table is constructed in main storage by a file initialization routine which reads the track indices sequentially from each track. To conserve storage space, the table is formed of halfword header ID increments which range in value from zero to $(2^{15}-1)$.[8] Should header ID's vary by more than $(2^{15}-1)$ over a single track, the table entry is expanded to a fullword which will accomodate increments as large as $(2^{31}-1)$. The end of a track is marked by a fullword entry having a value of zero.[9]

This technique for locating the track wherein a record may be found works properly in all cases except one. The exception occurs where detail records belonging to some one header record overflow into the track beyond that containing their header. Here, the "found" track will always be the second of the two tracks. If the record sought is a detail which falls on the second track, no problem exists. However, if the detail record sought is on the first of the two tracks, the "found" track is obviously not the correct one. This condition is discovered as soon as the track index is examined, and when it occurs, the track number is decremented to give the correct track.

---

8 These increments, unlike those contained in the halfword prefixed to each logical record, are cumulative. (See fn. 5, page 12.)

9 This is an "illegal" entry, as an increment of zero would normally be stored in a halfword.

FILE UPDATING

The deletion of logical records from a file is relatively simple: the record to be deleted is accessed, the Record Status Code[10] is suitably altered to indicate that the record has been deleted, and the record is written back into its place in the file. In this fashion, a record is effectively deleted from a file by making it inactive. Inactive records are physically deleted from each file on a periodic basis using a file reorganization program.

Adding new logical records to a file is considerably more complex. First, since "old" logical records are packed contiguously within the physical extent of a file, it is impossible to insert new records in their physical midst. So, space is allotted at the end of each data set, as it is created on disk, for additions. Second, with this arrangement for adding new records to a file, physical order no longer implies logical order. This problem is solved through the use of a system of flags and pointers which "connect" each record to the next in logical sequence.

Now, suppose a file contains records *A*, *B*, *C* and *D*, in that logical and physical order, and record *E* is to be inserted in logical sequence between *B* and *C*. The sequence of operations is as follows.

First, the one-bit addition flag in the halfword control section prefixed to *B* must be set "on" to indicate that a record has been added after *B*.

Second, a pointer is set up to link *B* to *E*, since *E* will be stored at the end of the file. Now, because there should be relatively few out-of-sequence records in each file,[11] no space has been provided for such pointers in the data

---

10 The Record Status Code is a four-bit data element included in the format of each logical record in every file.

11 The reason for there being relatively few out-of-sequence records in each file is that the files are reorganized on a daily basis at the time the System Backup File is created. For a complete description of this process, see Section 4.1.2

sets proper. Instead, a special file (the Additions File) has been set up to contain the pointers for all out-of-sequence records for every direct access data file in the system. Additions File entries have the following format: a fullword ID and a fullword pointer of the form TTRD. In the example under consideration, the fullword ID consists of one byte containing the Data Set Reference Number of the file, and three bytes containing a number which is equal to the displacement (in bytes) of B from the beginning of the main file; the pointer contains the address of E as follows: two bytes for the track number, one byte identifying the physical record containing E within the track, and one byte containing the displacement in bytes (within the physical record) of the first byte of E.

Displacement of B	Track Nbr. (2 bytes)	Phys. Record (1 byte)	Displ. of E (1 byte)
	T T	R	D
◄— Entry ID —► (fullword)	◄— Pointer	(fullword)	—►

Figure 5. Format of an entry in the Additions File.

Third and last, the logical record E, together with some control information, is placed at the end of the file. The record which contains E is unlike the standard logical record with its halfword prefix; its format consists of a fullword pointer of the form TTRD to C, another fullword containing the header or detail ID of E (the first bit of the fullword indicates whether the ID is for a header record or detail record), and n bytes of data E itself).

T T R D		
◄ POINTER TO C ► (fullword)	◄ HDR/DTL ID OF E ► (fullword)	◄ RECORD E ► (n bytes)

Figure 6. A new record and its prefixed control information.

The end result is a physical sequence of *A, B, C, D, E,* and a logical sequence of *A, B*→*E*→*C, D.*

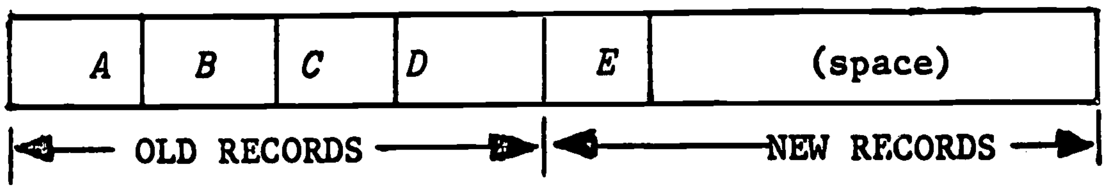

Figure 7. Schematic representation of a file showing a new record.

The only other situation which remains to be considered occurs when two new records fall logically adjacent to one another. So, suppose that record *F* is to be inserted logically between *E* and *C*. Here, the pointer prefixed to *E* is modified such that it contains the address of *F* (which is put at the end of the file), and the pointer prefixed to *F* is set to the address of *C*. Thus, the physical sequence of the records will be *A, B, C, D, E, F,* and the logical sequence will be *A, B*→*E*→*F*→*C, D.*

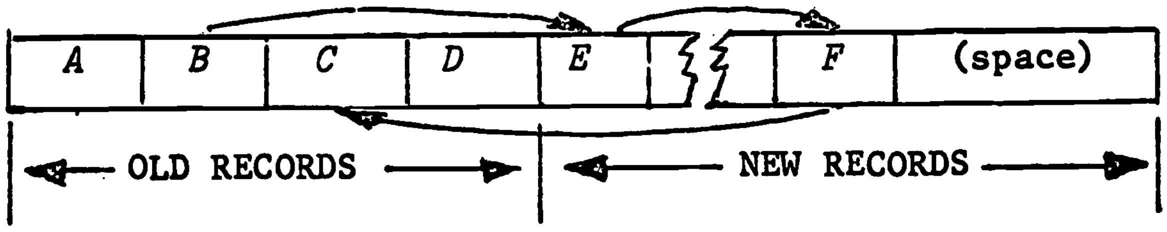

Figure 8. Schematic representation of a file showing two new logical records which are logically adjacent, and the pointers (designated by arrows) 'connecting' the records in logical sequence.

This scheme of adding new records to a file modifies the lookup scheme for finding a given record in a file. The complete lookup algorithm is presented here as a sequence of nine steps:

(1) Locate the track containing the record sought by looking up the header ID in the track table

(2) Locate the physical record through a header/detail ID lookup in the track index for that track

(3) Search through the physical record for a header/detail ID match with the ID of the record sought; if a match is found, exit from the lookup routines

(4) If a match is not found, check the one-bit addition flag in the halfword prefix to the logical record immediately

preceeding (in physical sequence) the record sought

(5) If this bit is not "on", exit under the "record not found" condition

(6) If the bit is "on", locate the appropriate Additions File entry; if no entry is found, an "impossible" error has occurred - the program terminates abnormally and the contents of main storage are printed out[12]

(7) If an Additions File entry is found, get the address of the added record

(8) Scan the "additions" portion of the main file to locate the added record; if the record is found, exit from the lookup routines

(9) If the added record is not found, exit under the "record not found" condition

3.2 Sequential-Access Files

As mentioned above, all sequential-access files, whether stored on disk or magnetic tape, are generated and accessed using either BSAM or QSAM as provided by OS/360. The logical and physical organization of these files, then, conform to the stock specifications for these access methods. For a complete description of BSAM and QSAM, the reader is referred to the following IBM Systems Reference Library (SRL) manuals: IBM System/360 Operating System: Concepts and Facilities, Form C28-6535 (general treatment), and IBM System/360 Operating System: Data Management, Form C28-6537 (full description of both BSAM and QSAM).

12 Only two kinds of errors can cause the abnormal termination of the program at this point: a programming error, or a system failure where different portions of the files have been partially updated to different degrees.

4. FILE SPECIFICATIONS

The files associated with the Network's system are divided conceptually into three groups: the system files, input/output files, and data files. The system files is that group of files which contain the system's programs, as well as the system backup files. The input/output files contain tables which are used in I/O editing and similar operations, and also include files which are used to temporarily store actual input and output records. The data files form the true data base for the system - information about films, libraries, customers, and bookings.

Prior to presenting the discussion of the system's files, some comments on the tables used to describe the format of the records in each file are in order. These tables consist of six columns which contain various categories of information about each field in the record being described. The labels of these columns, and some notes on their contents, are as follows:

FIELD LABEL - This column contains a seven-character label which uniquely identifies the field described. Each label has two parts: the first two characters identify the data file record to which the field belongs, and the last five characters are a universal mnemonic which applies to the field wherever it is located. Thus, for example, the label 'DCCUSTM' identifies the Network customer number ('CUSTM') in the detail record of the Short Catalog/Bookings File ('DC'). Similarly, 'DECUSTM' identifies the same field in the Customer Information File. It should be recognized that these labels may or may not be used by the programs handling the records; they are primarily a convenient means for identifying each field in the documentation of the system.

FIELD CONTENT - This column contains a short verbal description of the content of the field.

FIELD ATTRIBUTES - Data may be stored internally in any one of three "formats" in the S/360 computer: alphanumeric (character), packed decimal, or binary.[13] The Network's system utilizes only two of these forms of representation - alphanumeric for fields which may contain alphabetic as well as numeric characters, and binary for wholly numeric fields. The content of

13 For a more complete discussion of S/360 data representation, see Appendix B.

this column indicates which type of representation is used for the field being described.

NBR. CHAR. - This column contains one measure of a field's length - its number of characters. The figure which appears in this column is always relative to the type of S/360 representation used for the field. That is, if the field is stored as alphanumeric, the number refers to alphanumeric characters (each alphanumeric character is stored in one full byte), whereas if the field is stored as binary, the number refers to binary characters or digits (eight binary digits are stored in one byte).

NBR. BYTES - This is another means of measuring a field's length: this column shows the number of bytes allotted each field. In most cases, a field requires a length of one, two, four, or more bytes. However, in some instances, several short numeric fields of one, two, or four bits are packed together into a single byte or, more usually, a halfword. In the latter case, rather than indicate in this column the fractional part of the byte occupied by the field, the whole group of fields is designated as being stored in the byte or halfword by the use of a bracket (]).

FIELD NBR. - The number in this column is similar to the field label in that it uniquely identifies the field within its file. These field numbers are used almost exclusively in (a) the editing and formatting transaction input and output records, (b) describing the contents of file records, and also (c) some file maintenance operations.

The fields comprising the file record described by the table are arranged vertically in sequence from first to last. Unused bits and bytes are noted where they occur in the record.

4.1 System Files

4.1.1 Program Load Module File

OS/360 CATALOG NAME: FILMLIB.EX

STORAGE MEDIUM: 2314 disk

ACCESS METHOD: Partitioned Access Method (PAM)[14]

FILE CONTENT: This file contains all the various processing programs, routines and subroutines of the Network's system, except those which, even though they are utilized by the system, are a part of OS/360 proper. Included is the system's Executive or nucleus which consists of certain error routines, a program caller, the system statistics updating routine, and the BTAM program.

FILE USE: During all phases of operation, the system resides in a 43,008-byte upper partition of main storage. Of this available storage, approximately 20K bytes is required by the system's Executive at all times: the Executive is loaded by OS/360 at operator command at the beginning of each day's operations, and is not released until the day's operations have been concluded. The remaining 23K bytes of main storage must be utilized by the system's other processing programs and their associated tables and communications sections. As a result of the limited storage available, the programs and tables are stored on disk and are called into the partition only when they are required. This file contains all the system's programs in load module form.

NOTES: The exact size and content of this file is determined by the programs of the system.

4.1.2 System Backup Files

OS/360 CATALOG NAME: FILMLIB.BACKUP

STORAGE MEDIUM: Magnetic tape

ACCESS METHOD: (Sequential-Access) EXCP

FILE CONTENT: There exist, at all times, three System Backup Files which are stored individually on three magnetic tapes. Each file consists of a

---

14 PAM is an OS/360 access method; its description may be found in IBM's SRL manual IBM System/360 Operating System: Concepts and Facilities (Form C28-6535).

complete copy of all the direct access files in the system, as they appeared at one point in time. Such a copy is said to be a generation of the files, and each of the tapes is identified as containing generation (0), (-1) or (-2), where (0) is the most recent and (-2) the oldest. The tapes are rotated such that the one containing the oldest generation is always used in the creation of the next (newest) generation.[15]

FILE USE: The System Backup Files serve three purposes: File generation routines use the most recent generation of the Backup Files to recreate all direct-access files at the beginning of each day's operations. Also, a by-product of producing the Backup Files at such frequent intervals is the correspondingly frequent reorganization of all the direct-access files in the system, which results in increased system efficiency. However, the principal purpose of the Backup Files is to provide a backup for all the direct-access files in the event of a drastic error by the system's programs, the computer's hardware, or the computer's software.

FILE GENERATION: A System Backup File is created each day at the end of the Network's operation. It is generated by (a) opening each direct-access file in sequence, (b) reading the entire file from beginning to end, while (c) formatting the logical records from the file into uniform blocks of 2048 bytes, which are (d) written out on the Backup File tape. During this operation, each direct-access file is reorganized such that the physical sequence of the records corresponds exactly to their logical sequence. In other words, any physically out-of-sequence records (those added to the file since the last reorganization and consequently stored at the end of the file) are restored to their 'proper' physical sequence within the file.

FILE FORMAT:

(1) The first physical record may contain a system label. (This is optional.)[16]

15 There is a fairly comprehensive discussion of the notion of data set generations in the IBM SRL manual IBM System/360 Operating System: Data Management, under "Data Set Control Facilities".

16 For a complete description of tape-stored data set labelling and organization, see the section "Data Set Storage and Volumes" in the IBM SRL manual IBM System/360 Operating System: Data Management.

(2) Each file loaded onto the System Backup File tape is stored in *n* 2048 byte physical records, as follows:

(a) First 2048 byte record -

bytes 0-1 - contains the value (-1) which marks the beginning of a file;

bytes 2-3 - contains the data set reference number;

bytes 4-7 - contain the length of header records in the file;

bytes 8-11- contain the length of detail records;

bytes 12-2047 - contain successive logical records, each with a ten byte prefix where the first halfword of the prefix contains the displacement (relative to the beginning of this record) of the next logical record, and the remaining doubleword contains the record ID of this record.

(b) (*n*-2) successive 2048 byte records -

bytes 0-2047 - contain more logical records.

(c) Last 2048 byte record -

bytes 0-1 - contain the value (-1) which marks the end of a file.

(3) The end of the Backup File is denoted by a tape mark.

4.2 Input/Output Files

The four input/output files associated with the Network's system are of two kinds. Two of the files, the Input File and the Output File, are used to temporarily store input records received under the batched mode of operation,[17] and output records which are to be emitted in either the batched or telecommunications mode of operation, respectively. The other two files contain tables which describe the formats of input messages, internal input records, data file records, internal output records and output messages; these tables are used by various programs which perform I/O editing functions and file updating functions.

In this document, only the format and content of the latter kind of I/O files are presented in detail. The use of these tables is complex, and the reader is referred to the detailed discussion of their use in the SFLN manual SFLN: System-1 Specifications - Input and Output (Doc. #SD-004-0) under the sections "Input Processing" and "Output Processing".

17 See the SFLN manual, SFLN: System-1 Specifications - Inputs and Outputs (Doc. #SD-004-0), for a discussion of the batched mode of operation.

4.2.1 Input File

OS/360 CATALOG NAME: FILMLIB.INPUT

STORAGE MEDIUM: 2314 disk

ACCESS METHOD: (Sequential-Access) BSAM

FILE CONTENT: All transaction inputs submitted to the Network's system in the batched mode of operation are edited into internal input records and are stored in the Input File to await further processing.[18]

FILE USE: The file is used as a holding place for the internal input records until such time as the batched processing of the records is to be performed. Just prior to the actual processing of these transaction inputs, a sort is performed on the key information (the sort key fields) of all the records in the file, in order to properly sequence the records for the processing programs.

RECORD FORMAT: Each internal input record consists of four fullwords (16 bytes) of key information and six fullwords (24 bytes) of data, as follows:

FIELD LABEL	FIELD CONTENT	FIELD ATTRIBUTES	NBR. CHAR.	NBR. BYTES
	Sort Key Fields			
DFLIBNØ	Network library number	Binary	8	4
DFFILNØ	File number	"	8	
DFTRANC	Transaction code	"	8	
	[unused bits]		8	
DFSUBKY	Subkey field	"	8	2
DFPRØCR	Processor routine code	"	8	
DFINDAY	Input date	"	16	2
DFHKEY1	Header key field	"	32	4
DFDKEY1	First detail key field	"	16	2
DFDKEY2	Second detail key field	"	16	2
	DATA		24	

RECORD LENGTH: (40 bytes) 10 words

NBR. OF LOGICAL RECORDS:

STORAGE ESTIMATES:

---

18 The portion of the Network's system which performs batched mode operations has not been implemented.

SEQUENCING OF RECORDS: Prior to the pre-processing sort, the records are stored in this file in the order in which they are received by the system. After the sort, the sequence is by (1) Network library number, (2) file number, (3) transaction code, (4) header key field, (5) first detail key field, (6) second detail key field, (7) subkey field, (8) processor routine code, and (9) input date.

BACKUP SOURCES: The batched card input comprises the backup source for this file.

FILE LIFESPAN: Ordinarily, the lifespan of this file should be fairly short - probably less than one day.

4.2.2 Output File

OS/360 CATALOG NAME: FILMLIB.ØUTPUT

STORAGE MEDIUM: 2314 disk

ACCESS METHOD: (Sequential-Access) BSAM

FILE CONTENT/USE: The Output File is used to temporarily store internal output records prior to their being emitted via the Teletype terminal or the S/360 printer. Not all internal output records need to be stored in this fashion - many are emitted immediately after they are generated. There are, however, three categories of internal output records which must utilize this temporary storage: (a) all output records generated during the processing of transactions in the batched mode of operation (these must be sorted and emitted on the S/360 printer as a series of reports),[19] (b) output records generated from processing in the telecommunications mode those information display transactions which require that their output be computer-printed, and (c) output records from information display transactions that require their output to be sorted prior to being printed via the Teletype terminal (e.g., the Generate Shipping List transaction).

19 The reader is reminded that the portion of the system which handles batched mode operations has not been implemented at this time.

RECORD FORMAT:

FIELD LABEL	FIELD CONTENT	FIELD ATTRIBUTES	NBR. CHAR.	NBR. BYTES
	Sort Key Fields			
DGØMØDE	Output mode code	Alphanumeric	1	1
DGLIBNØ	Network library number	"	3	3
DGREPTC	Output report code	"	2	2
DGINDAY	Input date	"	7	7
DGØKEY1	First output key field	"	14	14
DGØKEY2	Second output key field	"	14	14
	DATA			80

RECORD LENGTH: (128 bytes) 32 words

NBR. OF LOGICAL RECORDS:

STORAGE ESTIMATES:

SEQUENCING OF RECORDS: The records are stored in the order in which they are received from the transaction processing programs. In the event that the records are to be sorted prior to being emitted, a sort is performed on the first seven fields (the sort key fields that are contained in the first 48 bytes of the record) after all the output records have been stored in the file, after which they are in proper order for printing the output reports.

BACKUP SOURCES: This file may be restored in one fashion only: by completely regenerating all the output records which must be accomplished by reprocessing the input records from which the output was originally generated.

FILE LIFESPAN: The ordinary lifespan of this file is relatively short - perhaps less than a few hours - and is dependent upon the length of time needed to process all the transactions for which output is directed to this file, in addition to any time required for sorting this file, and any time-lapse between completion of the sort and printing the output.

4.2.3 Main File Descriptor Tables File

OS/360 CATALOG NAME: FILMLIB.02

STORAGE MEDIUM: 2314 disk

ACCESS METHOD: (Direct-Access) EXCP

FILE CONTENT: This file contains the set of tables which describe the content and format of each of the system's data files against which transactions may be processed. There is one table for each such data file, and each table is composed of records which describe, in fairly complete terms, the fields and subfields that are in the data file record itself, and/or are data elements of an input or output message or record for some transaction that may be processed against the data file. In this fashion, a main file descriptor table describes both the format and content of the data file record and also (in effect) the format and content of the I/O of each transaction that is processable against the data file.

An example should clarify this.. Suppose that a data file associated with the system has records that contain the following fields: *A, B, C, D, E* and *F*. Further, suppose that the input of a transaction (call it *x*) contains fields *A* and *B*, and also *M* and *N* which are not found in the file records, and that the output of *x* contains *A, B, C* and *Ø*. Likewise, the input of another transaction, *y*, contains *A, B, C* and *P*, and its output has *A, B, D, E* and *Q*. The main file descriptor table for this data file, then, would consist of records describing the following fields and subfields: *A - F* and *M - Q*, or, all the fields and subfields that occur in the file record and also all those that occur in the I/O of the transactions pertaining to the file.

FILE USE: The data contained in the main file descriptor tables is used by the programs (EIØMØD and its member programs) which perform the various editing functions in reformatting both internal input records from input messages and internal output records from data stored in the data files, for all transactions processed against the data files. In addition, information contained in these tables is used by some of the programs which process the file maintenance transactions.[20]

RECORD FORMAT:

FIELD LABEL	FIELD CONTENT	FIELD ATTRIBUTES	NBR. CHAR.	NBR. BYTES	FIELD NBR.
EFILNUM	File number	Binary	8	} 2	001
EFLDNUM	Field number	"	8		002
EFLDNAM	Field name	Alphanumeric	7	7	010
EFLDLNG	Field length (input/char.)	Binary	8	} 3	011
EFLDBIT	Field length (intern./bits)	"	8		012
	[unused bits]		8		——
EINPØP	Input option flags byte	"	8	} 4	013-020
EØUTØP	Output option flags byte	"	8		021-026
EMANØP	Internal option flags byte	"	8		027-031
	[unused bits]		8		——
EFLDLØW	Field value - low limit	"	16	2	032
EFLDHI	Field value - high limit	"	16	2	033
EINPSPL	Input special proc. code	"	8	} 2	034
EØUTSPL	Output special proc. code	"	8		035
EINTSTR	Input record start bit	"	16	2	036
EFILSTR	File record start bit	"	16	2	037
EØUTSTR	Output record start bit	"	16	2	038
	[unused bits]		32	4	——

RECORD LENGTH: (32 bytes) 8 words

NBR. OF LOGICAL RECORDS: 120 (approx.)

STORAGE ESTIMATES: 3840 bytes, exclusive of control information.

(Calculation parameters: 120 records, 32 bytes per record.)

SEQUENCING OF RECORDS: (1) File number

(2) Field number

BACKUP SOURCES: The primary backup source is prior generations of the file stored in the System Backup Files; secondary backup may be provided by the transaction input data used to create the tables.

FILE LIFESPAN: The lifespan of the file is more or less indeterminate, although of long duration: the file will not change unless new data files or

20 For a complete description of the editing processes, see SFLN Doc. #SD-004-0 (mentioned above) under the sections "Input Processing" and "Output Processing". For descriptions of the use of these tables in file maintenance transaction processing, the reader is referred to the same document and the section on "File Maintenance Transactions".

transactions are added to the system, or unless existing data files and/or transaction I/O messages and records are modified.

NOTES: Most of the content of the records in this file is straight-forward in terms of how it is presented in the RECORD FORMAT - Subsection above. However, the three option flag bytes (EINPØP, EØUTØP and EMANØP) contain many flags that are not described in that subsection, and so their contents are presented here:

EINPØP:

FIELD LABEL	FIELD CONTENT	FIELD ATTRIBUTES	NBR. BITS
IALPHA	Field is alphanumeric flag	Binary	1
IØPTNL	Field is optional on input flag	"	1
INØINP	Field is not inputted flag	"	1
INØMØV	Field is not moved on input	"	1
IEQUAL	Equality value check	"	1
IRANGE	Range value check	"	1
IPRECV	Pre-conversion spl. proc.	"	1
IPSTCV	Post-conv. special process'g	"	1

EØUTØP:

FIELD LABEL	FIELD CONTENT	FIELD ATTRIBUTES	NBR. BITS
ØLSUBF	Last subfield in string flag	Binary	1
	[unused bits]		2
ØNØMØV	Field not moved on output	"	1
ØSUBFL	Field is a subfield	"	1
ØREGSL	Field fetch register select	"	1
ØPRECV	Pre-conv. special processing	"	1
ØPSTCV	Post-conversion spl. proc.	"	1

EMANØP:

FIELD LABEL	FIELD CONTENT	FIELD ATTRIBUTES	NBR. BITS
	[unused bit]		1
MNØFIL	Field is not in file flag	Binary	1
MHDRKY	Field is header record key	"	1
MDTLKY	Field is detail record key	"	1
MDYNMC	Dynamic bit	"	1
MCALTB	Call table to cross-ref. field	"	1
	[unused bits]		2

Basically, these bytes containing flags act as sets of switches which govern certain functions of field editing that are performed by the EIØMØD program group during the processing of input and output.

One additional aspect of this file needs discussion, and it concerns the identification and sequencing of the records within a table. It is convenient for the system users and system maintenance personnel to be able to identify the various subfields and fields comprising a table with respect to whether they are key fields in data file records, data fields in data file header or detail records, or fields which are found only in the I/O of transactions processed against the file. This kind of identification is provided by means of the field numbers (EFLDNUM) assigned to the subfields and fields; the numbers are assigned in such a fashion that they comprise a classification scheme, which is as follows:

FIELD NBR. GROUP	FIELD CLASSIFICATION
01-05	Key fields in header records
06-09	" " " detail "
10-49	Data fields in header records
50-69	" " " detail "
70-99	Fields in transaction I/O units

Thus, for example, if a header record in some data file had three keyfields and ten data fields, the key fields would be assigned numbers 01-03 and the data fields numbers 10-19. If detail records from the same file had two key fields and fifteen data fields, the key fields would be numbered 06-07 and the data fields 50-64. Any other fields (presumably those found in transaction inputs and outputs and not in the header or detail records) would receive field numbers in the 70-99 range.

4.2.4 Character Record Maps File

OS/360 CATALOG NAME: FILMLIB.03

STORAGE MEDIUM: 2314 disk

ACCESS METHOD: (Direct-Access) EXCP

FILE CONTENT: Character record maps are keys to the format of two units of transaction I/O: telecommunications input messages and internal output records; a single map consists of a group of records which describes either a single input message or internal output record. The structure of a map is relatively simple: each record in the group is identified by a Network library number, a file number, a transaction number, and a record sequence number, and each record contains only one item of data - the number of a main file descriptor table entry (the field number, EFLDNUM). The first three identification numbers are constant within a group, and serve to identify the map; the record sequence numbers are assigned within each group so that the sequence of the fields designated in the map records corresponds exactly, by virtue of the order of the map records, to the sequence of the subfields and fields comprising the I/O unit described. Finally, the kind of I/O unit described by the map is indicated by a code stored as a part of each map record.

FILE USE: The character record maps are used by some of the programs from EIØMØD which perform two of the principal transaction I/O editing functions: the reformatting of internal input records from telecommunications input messages, and the generation of internal output records from data stored in data files and program work areas and communications sections in main storage. The manner in which the maps are utilized is relatively straightforward. In both cases, the number of the Network library submitting the transaction, the number of the file against which the transaction is being processed, the transaction number and the I/O unit code (input message or internal output record) are used to locate the appropriate map in the character record map file. Next, the file and field numbers (the latter is the data item stored in the map records - EFLDNUM) from the map records are used to locate the main file descriptor table entries for the fields indicated by the map. Finally, the information contained in the table entries is used by the editing programs for a field-by-field editing process which transforms the input message into the internal record (in the case of input processing) or which generates the internal output record from the stored data (in the case of output processing).

RECORD FORMAT: Not all the data stored in a character record map record is stored within the record proper: the four major fields identifying the map (library number, file number, transaction number and I/O unit code) are stored in the doubleword record ID which is prefixed to each logical record in any direct-access file. The remaining key field (the record sequence number) and the field number (EFLDNUM) are the two data items stored in the doubleword map record proper. Thus, the format of records in this file is as follows:

← prefixed record ID →				← map record proper →			
LIBRARY NBR.	FILE NBR.	TRANSACT. NBR.	I/O UNIT CODE	unused	REC. SEQ.	unused	FLD. NBR.
← 8 bytes →				← 8 bytes →			

RECORD LENGTH: (8 bytes) 2 words

NBR. OF LOGICAL RECORDS: 520 (approx.)

STORAGE ESTIMATES: 4160 bytes, exclusive of control information.

(Calculation parameters: 520 records, 8 bytes per record.)

SEQUENCING OF RECORDS: (1) Network library number

(2) File number

(3) Transaction number

(4) I/O unit code

(5) Record sequence number

BACKUP SOURCES: Primary backup is provided by the System Backup Files; secondary backup may be provided by the transaction input data used to create the maps.

FILE LIFESPAN: This file will be altered only when transaction inputs are altered, or when new transactions are added to the system or current transactions are dropped. The file must be in existence if the system is to function.

NOTES: One important aspect of this file bears mention. Since the maps are identified by library number as well as by file and transaction number (these two fields comprise the transaction code), the I/O of any transaction in the system may be tailored to the needs of each particular library. While there are limits on the disparities between the I/O for a given transaction for, say, two different Network libraries, it is possible to effect some economy in telecommunications transmission costs by eliminating unneeded or unused subfields and fields by simply specifying different character record maps for different libraries within the Network. In addition to contributing to better economies in telecommunications, this factor also contributes to the general flexibility of the Network's system.

4.3 Data Files

The data files associated with the system contain four basic categories of information: information about films, bookings made on the films, the Network's member libraries, and the libraries' customers. Two of the files contain information about films: the Master Catalog File and the Short Catalog & Bookings File; the latter file also contains records of all the bookings made on the films. Information about the Network libraries is contained in the Film Library Tables File, and the Customer Information File contains data on the libraries' customers. Two other files fall into the category of data files; the first is the Film Number Cross-Index File which is an inverted index file of film numbers, and the second is the Additions File which contains pointers to records added to all the direct-access files in the system.

4.3.1 Master Catalog File

OS/360 CATALOG NAME: FILMLIB.01

STORAGE MEDIUM: 2314 disk

ACCESS METHOD: (Direct-access) EXCP

FILE CONTENT: This file serves as a master catalog of materials in the collections of the Network's member libraries. As such, the file contains a single record for each unique film within the Network; each of these records currently contains some bibliographic information about the film, together with information pertaining to its intra-Network use.

Since film use information properly belongs in the header record of the Short Catalog & Bookings File, it is anticipated that the use information currently contained in this file will be transferred eventually to such header records. It is also anticipated that the bibliographic information now contained in the Master Catalog File will be greatly expanded in the future, and that various kinds of indexing information (e.g., subject-classification) will be added to the file.

FILE USE: This file serves as a primary information source for materials contained in the collections of the Network libraries: it functions as a union-list of materials that are available within the Network. While the file currently contains only records of films, provisions have been made so that other forms of educational media may be represented.

RECORD FORMAT:

FIELD LABEL	FIELD CONTENT	FIELD ATTRIBUTES	NBR. CHAR.	NBR. BYTES	FIELD NUMBER
DASTITL	Short title	Alphanumeric	18	18	10
	[unused char.]	"	1	1	
DASUCAT	SUFRL catalog number	"	5	5	11
	[unused char.]	"	1	1	
DASUINV	SURFL inventory number	"	5	5	12
	[unused char.]	"	1	1	
DADISTR	Distributor code	"	3	3	13
	[unused char.]	"	1	1	
DAURENT	SUFRL rental	"	4	4	14
	[unused char.]	"	1	1	
DASUPRS	SUFRL print numbers	"	14	14	15
	[unused char.]	"	1	1	
DACØLPR	SUFRL nbr. color prints	"	2	2	16
	[unused char.]	"	1	1	
DABWHPR	SUFRL nbr. b+w prints	"	2	2	17
	[unused char.]	"	1	1	
DATRANN	Update transaction code	"	2	2	
DASRENT	Shared rental flag	"	1	1	18
	[unused char.]	"	1	1	
DABØCPR	Network library print nbrs.	"	21	21	19
	[unused char.]	"	1	1	
DAIDENT	Identification number	"	5	5	01
DAICHEK	Ident. nbr. check digit	"	1	1	20
	[unused char.]	"	27	27	21

RECORD LENGTH: (120 bytes) 30 words

NBR. OF LOGICAL RECORDS: 7000 (approx.)

STORAGE ESTIMATES: 840,000 bytes, exclusive of control information. (Calculation parameters: 7000 records, 120 bytes per record.)

SEQUENCING OF RECORDS: (1) Identification number

BACKUP SOURCES: Primary backup source consists of prior generations of the file on the System Backup Files; secondary backup would consist of card or punched paper tape input used to create and/or update the file.

FILE LIFESPAN: Long.

4.3.2 Short Catalog & Bookings File

OS/360 CATALOG NAME: FILMLIB.17

STORAGE MEDIUM: 2314 disk

ACCESS METHOD: (Direct-Access) EXCP

FILE CONTENT: The information in this file is stored in both header and detail records. A header record contains information about a film which is pertinent to the scheduling of that film's usage. (As such, the set of header records may be considered to be an abbreviated catalog file.) A detail record contains information about one film booking.

Since information about a film varies from library to library, as well as from film to film, there will be one header record for each unique film/library combination. Detail records representing bookings of a specific film in a particular library fall immediately after the header record for that film/library combination.

FILE USE: Information contained in this file is used primarily in processing booking requests. The header records provide the booking program with constant information about each film, while the detail records contain information about the bookings that have been made on each film. The header records also serve a secondary function as a limited catalog source, and the detail records have various secondary uses as a source of information about bookings.

HEADER RECORD FORMAT:

FIELD LABEL	FIELD CONTENT	FIELD ATTRIBUTES	NBR. CHAR.	NBR. BYTES	FIELD NBR.
DBLIBNØ	Network Library number	Binary	8		001
DBMEDIA	Media code	"	8	4	002
DBIDENT	Identification number	"	16		003
DBSHELF	In-library shelf number	"	16		010
DBSCHEK	Shelf number check digit	"	4	4	011
DBDISTR	Distributor code	"	12		012
DBALPHA	Title alpha sequence number	"	24	4	013
DBSUBJK	Subject classification code	"	8		014
DBNBRPR	Total number of prints	"	4		015
DBNØBWH	Number of b&w prints	"	4		016
DBNØCØL	Number of color prints	"	4	2	017
DBREELS	Number of reels per print	"	4		018
DBRENTB	Rental of b&w prints	"	16	2	019
DBRENTC	Rental of color prints	"	16	2	020
DBSTITL	Short title	Alphanumeric	18	18	021
DBSEASL	Use seasonality code	Binary	4	2	022
DBDENSY	Use density table	"	12		023
DBSHARE	Use sharing code	"	4		024
DBSSHIP	Standard shipping code	"	4		025
DBLEASD	Leased film flag	"	1		026
DBSRENT	Shared rental film flag	"	1	2	027
DBSTATE	State-backup-has-it flag	"	1		028
DBREGNL	Regional-backup-has-it flag	"	1		029
DBHSTAT	Header Record Status code	"	4		030

DETAIL RECORD FORMAT:

FIELD LABEL	FIELD CONTENT	FIELD ATTRIBUTES	NBR. CHAR.	NBR. BYTES	FIELD NBR.
DCSHIPD	Shipping date	Binary	16	2	006
	[unused bits]	"	4	}	
DCPRINT	Print number	"	4	} 2	007
DCBDISP	Booking displacement	"	4	}	050
DCSHIPC	Shipping code	"	4	}	051
DCLASTD	Last unavailable date	"	16	2	052
DCDYBKD	Date booking was made	"	16	2	053
DCCUSTM	Network customer number	"	32	4	
	[consists of:				
	CLIBR				054
	CDIST				055
	CSCHL				056
	CTCHR]				057
DCUSEPD	Booked Use period	"	8	}	058
DCUDISP	Use date displacement[21]	"	4	} 2	059
DCRDISP	Return date displacement[22]	"	4	}	060
DCREQCD	Booking request code	"	4	}	061
DCBKDCD	Booking results code	"	4	}	062
	[unused bits]		1	}	
DCSHIPL	Shipping list flag	"	1	} 2	063
DCRECHG	Record change flag	"	1	}	064
DCØMØDE	Output mode code	"	1	}	065
DCDSTAT	Detail Record Status code	"	4	}	066

RECORD LENGTHS: Header record - (40 bytes) 10 words
Detail record - (16 bytes) 4 words

NBR. OF LOGICAL RECORDS: Header records - 5000 (approx.)
Detail records - [highly volatile - no estimate]

STORAGE ESTIMATES: The amount of storage required for the header records in the file is approximately 200,000 bytes (5000 records, 40 bytes per record). The amount of storage required for detail records is not feasible to calculate because the number of detail records contained in the file varies considerably from day to day.

SEQUENCING OF RECORDS: Header records - (1) Network library number
(2) Media code
(3) Identification number

Detail records - (4) Shipping date
(5) Last unavailable date

21 The displacement factor (number of days) forward from the shipping date (DCSHIPD) to the booked use date.

22 The displacement factor backward from the last unavailable date (DCLASTD) to the customer return date.

BACKUP SOURCES: Backup for this file is provided by prior generations of the file stored in the System Backup File.

FILE LIFESPAN: Complete turnover of this file should occur in a maximum of twelve months.

4.3.3 Film Library Tables File

OS/360 CATALOG NAME: FILMLIB.07

STORAGE MEDIUM: 2314 disk

ACCESS METHOD: (Direct-Access) EXCP

FILE CONTENT: A great variety of variables and parameters which pertain to each Network library are stored in this file. Generally, the content is divided into that which directly describes the library's booking process, i.e., various time parameters and delivery schedule tables, and that which more generally describes the library's operations. According to these criteria, the data is organized into header records (general information) and detail records (booking parameters and delivery tables). Both sorts of records are extremely large compared with most others in the system's files, and are therefore referred to as "library tables" and "delivery tables", respectively.

FILE USE: The header records in this file are used for a variety of purposes: to store daily and cumulative statistics on the library's operations, as storage for parameters and variables needed by various processing routines (e.g., for the Display Shipping List transaction), and also as storage for information required by various routines in the processing of requests for film usage. Detail records are used almost exclusively by the booking routines.

HEADER RECORD FORMAT:

FIELD LABEL	FIELD CONTENT	FIELD ATTRIBUTES	NBR. CHAR.	NBR. BYTES	FIELD NBR.
DGLIBNØ	Network library number	Binary	32	4	001
DGCURLN	Current input line number	"	16	2	010
DGTDATE	Date of last update to table	"	16	2	011
DGRLBNØ	Regional backup library nbr.	"	16	2	012
DGRLEAD	Lead time to regional lib.	"	16	2	013
DGRRETN	Return time to regional lib.	"	16	2	014
DGSLEAD	Lead time to state backup lib.	"	16	2	015
DGSRETN	Return time to state lib.	"	16	2	016
DGFWRIT	(contains 16 one-bit switches for file dumps as requested by library)	"	16	2	017
DGFWNRM	Normal settings for file-write switches	"	16	2	018
DGFSTDY	Shipping list-first date	"	16	2	019
DGLSTDY	Shipping list-last date	"	16	2	020
DGFILML	Ident. nbr. of 1st film in lib.	"	16	2	021
DGSLTRN	Ship. list transaction code	"	16	2	022
	[unused bytes]			20	——
DGTABLS	Number of delivery tables	"	16	2	023
DGTTCNB	Capture Normal Bkg.-tran. code	"	16	2	024
DGTTRNB	Request Normal Bkg.-tran. code	"	16	2	025
DGTSTAT	Transaction statistics table	"		720	026
DGBEXCT	DBS[23] - exact day bkgs.	"	16	2	027
DGBALTE	DBS - alternate day bkgs.	"	16	2	028
DGBCEXT	DBS - closest to ext. day	"	16	2	029
DGBCALT	DBS - closest to alt. day	"	16	2	030
DGBMRGL	DBS - made at regional	"	16	2	031
DGBNRGL	DBS - refused at regional	"	16	2	032
DGBMSTA	DBS - made at state	"	16	2	033
DGBNSTA	DBS - refused at state	"	16	2	034
DGLNAME	Network library name	Alphanumeric	10	10	035
DGQNSET	Request code normal setting	Binary	8	1	036
DGGFLAG	GD flag (1st bit next byte)			1	037
	[unused bytes]			20	——

23 DBS = daily booking statistic.

DETAIL RECORD FORMAT: There are two types of delivery tables, based on the types of delivery schedules utilized by the Network libraries. Some libraries deliver and pick up on mixed days of the week, while others service their customers on a daily basis. Both types of schedules are represented in detail records which contain a 20-byte information section followed by 400 halfword entries which comprise the delivery table itself. Each one of the halfword entries represents a day of the school year and contains information about that day.

The format of the detail record is as follows:

FIELD LABEL	FIELD CONTENT	FIELD ATTRIBUTES	NBR. CHAR.	NBR. BYTES	FIELD NBR.
	Information Section Fields:				
DHDTABL	Delivery table number	Binary	32	4	006
DHCLEAD	Customer lead time	"	8	2	050
DHCRETN	Customer return time	"	8		051
DHCMUSE	Customer minimum use period	"	8	2	052
DHSLACK	Number of slack days	"	8		053
DHADVBK	Max. wks. for advance booking	"	8	2	054
DHRUDIS	Cust. receive-use date displ.	"	8		055
DHTTYPE	Type of delivery table	"	1	2	056
DHREGNT	Regents days are bkg. days	"	1		057
	[unused bits]		6		—
DHLMUSE	Min. use period - all custmrs.	"	8		058
DHENDYR	Last ship. date of school yr.	"	16	2	059
DHRDISP	Return date displacement	"	8	1	060
	[unused bits]		40	5	—
	Delivery Table:			800	—

Halfword entries in a fixed date delivery table have the following format:

FIELD LABEL	FIELD CONTENT	FIELD ATTRIBUTES	NBR. CHAR.	NBR. BYTES	FIELD NBR.
DHDDISP	Booking date displacement	Binary	8	2	061
	[unused bits]		3		—
DHVALDY	Valid booking day	"	1		062
DHRGNTS	Regents day	"	1		063
DHSHØRT	Day is Sat., Sun., short holiday	"	1		064
DHRETND	Customer return day	"	1		065
DHLSHPD	Library shipping day	"	1		066

The format for daily delivery table entries is as follows:

FIELD LABEL	FIELD CONTENT	FIELD ATTRIBUTES	NBR. CHAR.	NBR. BYTES	FIELD NBR.
	[unused bits]		11	2	—
DHVALDY	Valid booking day	Binary	1		067
DHRGNTS	Regents day	"	1		068
DHSHØRT	Day is Sat., Sun., short holiday	"	1		069
DHRETND	Customer return day	"	1		070
DHLSHPD	Library shipping day	"	1		071

RECORD LENGTHS: Header record -(824 bytes) 206 words
Detail record -(820 bytes) 205 words

NBR. OF LOGICAL RECORDS: Header records - 3
Detail records -21

STORAGE ESTIMATES: 19,692 bytes, exclusive of control information. (Calculation parameters: 3 header records and 21 detail records; 824 bytes per header and 820 bytes per detail record.)[24]

SEQUENCING OF RECORDS: Header records - (1) Network library nbr.
Detail records - (2) Delivery table nbr.

BACKUP SOURCES: System Backup Files

FILE LIFESPAN: Header records have no defined lifespan; the delivery tables contained in the detail records must be updated on a yearly basis.

---

24 These estimates are for the three participating BOCES libraries only: Erie County #1, Westchester County #1, and Suffolk County #3.

4.3.4 Customer Information File

OS/360 CATALOG NAME: FILMLIB.14

STORAGE MEDIUM: 2314 disk

ACCESS METHOD: (Direct-access) EXCP

FILE CONTENT: Information about every customer of each Network library will be stored in this file. The information is to be contained in two record formats: numeric information will be stored in header records (one for each customer), while customer address information will be stored in detail records (one per address line).

FILE USE: Primary use is in processing booking requests and preparing output from such processing; secondary use is as a limited customer information source.

HEADER RECORD FORMAT:

FIELD LABEL	FIELD CONTENT	FIELD ATTRIBUTES	NBR. CHAR.	NBR. BYTES	FIELD NBR.
DECUSTM	Network customer number [consists of:	Binary	32	4	
	CLIBR				001
	CDIST				002
	CSCHL]				003
DELCUST	Library customer number [consists of:	"	32	4	
	LLIBR				010
	LDIST				011
	LSCHL]				012
	Unused bits		8 }		
DELCHEK	Library cust. nbr. check digit	"	4 }	2	013
DEDTABL	Customer delivery table nbr.	"	4 }		014
DENYSC1	N.Y.S. school code (part 1)	"	16	2	015
DENYSC2	" " " (part 2)	"	32	4	016
DESEQNØ	Cust. alpha/geogr. seq. code	"	16	2	017
DELEADT	SUFRL shipping lead time	"	4 }		018
DEGRADE	Grade level code	"	4 }	2	019
DENPUBL	Non-public school code	"	4 }		020
DEHSTAT	Header Record Status code	"	4 }		021

DETAIL RECORD FORMAT:

FIELD LABEL	FIELD CONTENT	ATTRIBUTES	NBR. CHAR.	NBR. BYTES	FIELD NBR.
DFADDRS	Address line	Alphanumeric	22	22	050
DFADDCD	Address type code	Binary	4	2	006
DFADDLN	Address line number	"	4		007
	[unused bits]		4		
DFDSTAT	Detail Record Status code	"	4		051

RECORD LENGTHS: Header record - (20 bytes) 5 words
Detail record - (24 bytes) 6 words

NBR. OF LOGICAL RECORDS: Header records - 300 (approx.)
Detail records - [none]

STORAGE ESTIMATES: 6000 bytes, exclusive of control information.
(Calculation parameters: 300 header records and no details; 20 bytes per header record.)[25]

SEQUENCING OF RECORDS: Header records - (1) Network customer nbr.
Detail records - (2) Address type code
(3) Address line number

BACKUP SOURCES: Backup sources include prior generations of the file stored in the System Backup Files, as well as a cumulative punched card file used to create and update the disk file.

FILE LIFESPAN: This is an on-going file, with no fixed lifespan. It is expected that updating activity will be generally light through the school year, and that most updating will occur after the end of each school year (July through August).

---

25 Note that these storage estimates are only for the three BOCES libraries currently in the system.

4.3.5 Film Number Cross-Index File

OS/360 CATALOG NAME: FILMLIB.09

STORAGE MEDIUM: 2314 disk

ACCESS METHOD: (Direct-Access) EXCP

FILE CONTENT: This file is a cross-index in which two kinds of film identification numbers are linked. Various libraries in the Network choose to identify their films using their own scheme of shelf or catalog numbers, while the Network's system requires that each distinct film represented in the system's files be internally identified in a unique and universal fashion. The 'external' and 'internal' identification numbers used for these purposes are designated shelf numbers and identification numbers, respectively.

Since shelf numbers vary from library to library, this file is divided into subfiles - one for each Network library which utilizes shelf numbers. Within each subfile, the shelf number for each film in the library represented is linked to its appropriate identification number.

FILE USE: The file is used in processing certain transaction inputs: any time a shelf number is encountered in a transaction input message, it must be replaced by its corresponding film identification number in order for the system to be able to process the transaction. The identification number (again, a number which is internal to the system and uniquely identifies each film in the Network) is obtained by performing a lookup on the given shelf number, in the appropriate subfile of this file.

RECORD FORMAT: The records in each subfile make use of the technique for identifying logical records in direct-access files. The identification number for each film is stored in halfword which is the logical record proper. The shelf number (on which a lookup is performed) is stored as a 14-bit record ID increment in the halfword which is usually prefixed to a logical record in a direct-access file. (See pp. 10-11 for a complete discussion of these halfword prefixes.)

RECORD LENGTH: (2 bytes) 1/2 word

NBR. OF LOGICAL RECORDS: 700 (approx.)

STORAGE ESTIMATES: 1400 bytes, exclusive of control information.
(Calculation parameters: 700 logical records, 2 bytes per record.)

SEQUENCING OF RECORDS: (1) Network library number
(2) Shelf number

BACKUP SOURCES: Prior generations of this file are stored in the System Backup Files.

FILE LIFESPAN: This is an on-going file which has no well-defined lifespan.

4.3.6 Additions File

OS/360 CATALOG NAME: FILMLIB.00

STORAGE MEDIUM: 2314 disk

ACCESS METHOD: (Direct-Access) EXCP

FILE CONTENT: [See Section 3.1, Direct-Access Files - File Updating.]

RECORD FORMAT:

FIELD LABEL	FIELD CONTENT	FIELD ATTRIBUTES	NBR. CHAR.	NBR. BYTES	FIELD NBR.
DAENTRY	Record identification nbr.	Binary	32	4	001
DANWREC	Pointer to added record	"	32	4	005

RECORD LENGTH: (8 bytes) 2 words

NBR. OF LOGICAL RECORDS: ---

STORAGE ESTIMATES: ---

SEQUENCING OF RECORDS: (1) Record identification number

FILE USE: The file is used to contain pointers to all records added to direct-access files over a day's operations - it is a means for maintaining the logical sequence of records in those files. For a detailed description of the file use, see Section 3.1, Direct-Access Files - FILE UPDATING.

BACKUP SOURCES: No immediate backup is provided; new records will have to be resubmitted to the system if this file is destroyed.

FILE LIFESPAN: The lifespan of this file is equal to the time between consecutive reorganizations of the direct-access files in the system, in this case, about one day.

5. SUMMARY

Within the Network's system, then, there are three classes of files, all of which are accessed and maintained by the system's own programs:

The system files contain the programs that perform all the various processing functions in the system; included is the system's own operating system or nucleus. Also included in this group are the system's backup files which provide a reasonable means for recovery in the event of disastrous system (S/360 or Network) malfunctions.

The data files contain all the various kinds of information about the libraries, library customers, films, and bookings that are necessary to the functioning of the Network. The records in these files are described with respect to both format and content by a collection of tables, and consequently their formats are modified with relative ease.

The tables comprise the bulk of the third class of files - the I/O files. Again, they make possible truly flexible record formats and generalized I/O processing programs, so that there exists a real independence of such record formats and programs.

In conclusion, it should be pointed out that the real elegance of such independence lies in the fact that the record formats and I/O processing programs may be modified independently of one another. The capacity for such modification is of prime importance in a system which may be subject to further development and change.

APPENDICES

APPENDIX A: List of Files

Direct-Access Files

(All direct-access Files are stored on IBM 2314 disk storage.)

FILE NAME	OS/360 CATALOG NAME	DSRN*
Additions File	FILMLIB.00	00
Master Catalog File	FILMLIB.01	01
Main Field Descriptor Tables File	FILMLIB.02	02
Character Record Maps File	FILMLIB.03	03
Film Library Tables File	FILMLIB.07	07
Film Number Cross-Index File	FILMLIB.09	09
Customer Information File	FILMLIB.14	14
Short Catalog & Bookings File	FILMLIB.17	17

Sequential-Access Files

FILE NAME	OS/360 CATALOG NAME	STORAGE MEDIUM
System Backup File	FILMLIB.BACKUP	Tape
Input File	FILMLIB.INPUT	Disk
Output File	FILMLIB.ØUTPUT	"

Partitioned Access Files

FILE NAME	OS/360 CATALOG NAME	STORAGE MEDIUM
Program Source Decks File	FILMLIB.SØURCE	Disk
Program Library (load modules)	FILMLIB.EX	"

* Data Set Reference Number

APPENDIX B: Notes on System/360 Organization and Data Storage

Organization

The System/360 data storage facilities have as their fundamental unit of organization the byte which is made up of eight bits. For convenience in the manipulation of stored data, bytes are organized into a larger physical unit, called the word (or fullword), which is four bytes in length. Further, fullwords are broken into halfwords (two bytes) and may be combined into doublewords (eight bytes). These four units, then, are used to describe the physical structure of S/360 main storage.

An important concept of data storage in the S/360 is the notion of boundary alignment. Consider a block of S/360 main storage *n* bytes long, where the bytes are numbered *0 - n*. Figure A shows the physical structure of such a block:

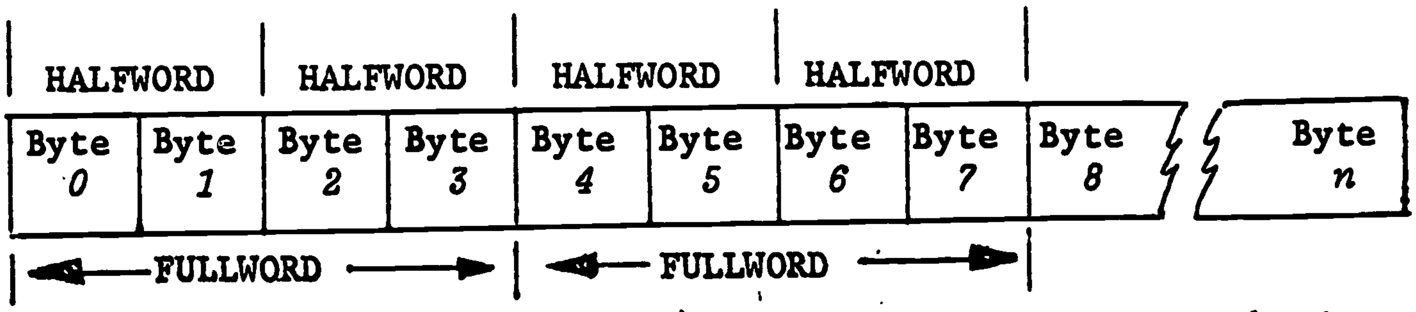

Figure A. Block of System/360 main storage *n* bytes long, showing physical organization.

Now, boundary alignment is simply the notion that any data element which is a halfword, fullword, or doubleword in length should be stored such that it falls within the boundaries of that particular unit. For example, a data element one fullword in length should be stored (if it is to be properly aligned) in, say, bytes *0 - 3* or *4 - 7* in Figure A. (Note that it would be possible, although extremely inconvenient from a programming point of view, to store the element in, say, bytes *1 - 4*.)[26]

26 A much more thorough description of these concepts is presented in Chapter 1 of IBM's Student Text, A Programmer's Introduction to the IBM System/360 Architecture, Instructions, and Assembler Language, IBM Form C20-1646.

Data Storage

Data stored in the S/360 may be represented in three forms, according to the nature of the data to be stored. If a field contains alphabetic characters or special characters (@, /, -, +, #, >, etc.), it must be stored in the character "format" or representation, where each character requires one byte of storage (EBCDIC code). Wholly numeric fields may be stored either in the packed decimal "format" (2 decimal digits per byte) or in the binary representation (8 binary digits per byte). Numeric data fields ordinarily contain a sign which occupies one digit position in packed decimal (i.e., 1/2 byte), and one or more positions (bits) in binary.[27]

27 See footnote 26, preceding page.

APPENDIX C: Notes on Disk Storage Devices

Disk storage devices are peripheral devices used for secondary storage in a computer system. The actual storage medium consists of a set of circular plates, covered with an oxide coating, which are mounted horizontally on a spindle and rotate at a high speed. Such a collection of plates on a spindle is called a disk-pack, and packs may or may not be permanently mounted on the disk drive unit, depending on the design of the unit. Data is stored and retrieved on the oxide surfaces by means of read and write heads (one set for each surface) mounted on arms that are usually moved as a group across the disk surfaces; the data is represented by magnetized spots or "bits". Since access to the data is cyclic due to the rotation of the pack, access is said to be "random" or direct, as distinguished from magnetic tape where access is linear or sequential.

Data storage on a disk-pack follows a specific pattern. Each surface is divided into tracks which are concentrically arranged bands or zones of data storage. Each surface is further divided into track segments which are arcs of a uniform length within each track (Figure A). The arms holding the read and write heads function as a unit, and all the heads are vertically parallel so that when the

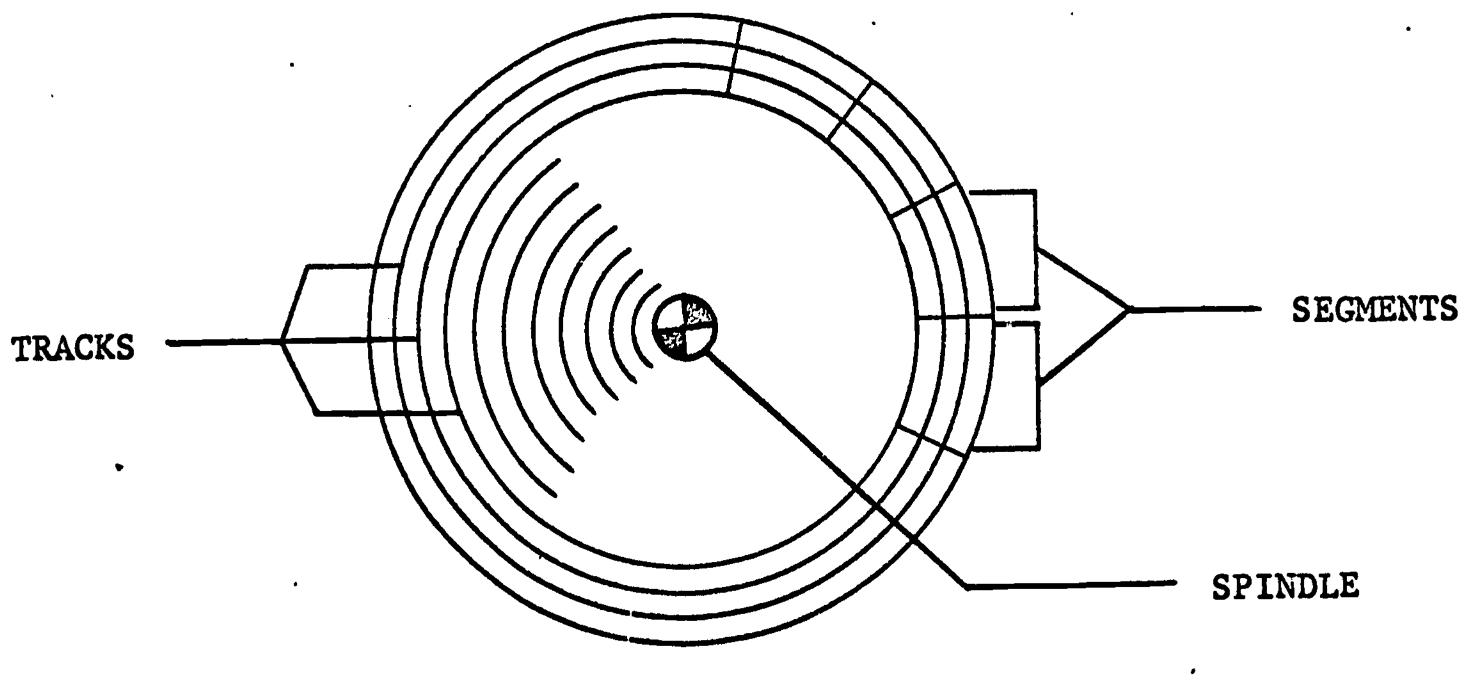

Figure A. Disk surface showing arrangement of tracks and segments.

heads on one arm are positioned to read and write a particular track on a given surface, all the other heads are similarly positioned to read or write on the corresponding tracks of the other surfaces. Due to this arrangement of the heads, all the tracks on a pack are grouped into cylinders: a cylinder consists of the set of tracks (one on each surface) that fall in the same vertical plane (Figure B).

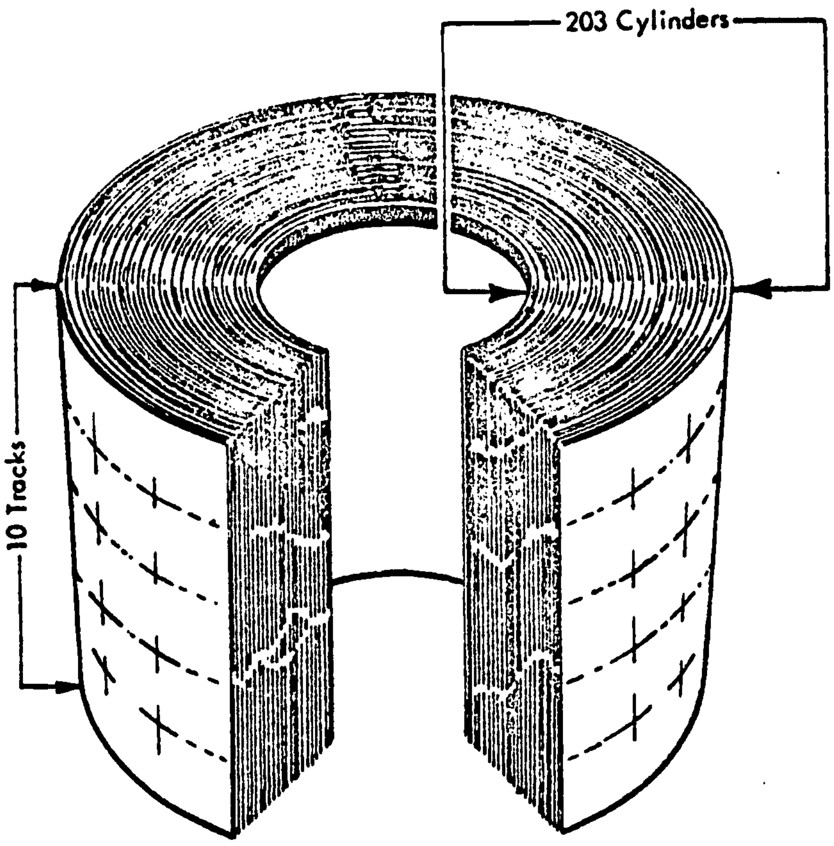

Figure B. Arrangement of cylinders in a disk pack (courtesy of IBM).

Data stored on disk, then, is stored in segments within tracks that are within cylinders. In OS/360 terminology, a single pack is referred to as a volume, and the data sets or files stored on disk may occupy any number of tracks or cylinders of a pack, or may even extend over several volumes.

BIBLIOGRAPHY

IBM publications:

Student Text materials -

A Programmer's Introduction to the IBM System/360 Architecture, Instructions, and Assembler Language, Form C20-1646

Introduction to IBM System/360 Direct Access Storage Devices and Organization Methods, Form C20-1649

Systems Reference Library materials -

IBM Operating System/360: Introduction, Form C28-6534

IBM Operating System/360: Concepts and Facilities, Form C28-6535

IBM System/360 Operating System: Data Management, Form C28-6537

IBM System/360 Operating System: Basic Telecommunications Access Method, Form C30-2001

SFLN publications:

Statewide Film Library Network: System-1 Specifications - Inputs and Outputs, Doc. #SD-004-0

121 COLLEGE PLACE · SYRACUSE, NEW YORK 13210