DOCUMENT RESUME

ED 061 978 52 LI 003 648

AUTHOR	Aiyer, Arjun K.
TITLE	The CIMARON System: Modular Programs for the Organization and Search of Large Files. Final Report.
INSTITUTION	California Univ., Berkeley. Inst. of Library Research.
SPONS AGENCY	Office of Education (DHEW), Washington, D.C. Bureau of Research.
BUREAU NO	BR-7-1083
PUB DATE	Sep 71
GRANT	OEG-1-7-071083-5068
NOTE	60p.;(6 References)
EDRS PRICE	MF-$0.65 HC-$3.29
DESCRIPTORS	*Bibliographic Citations; Computer Programs; *Electronic Data Processing; *Information Retrieval; On Line Systems; *Search Strategies
IDENTIFIERS	Berkeley; *University of California

ABSTRACT

The File Organization Project has made available a set of programs which are designed to operate on large files of machine readable bibliographic records. These programs are designed as an instrument for understanding and refining the techniques of bibliographic search. This document discusses four aspects of the system: (1) The retrieval program, CIMARON, is an on-line, interactive system with two complementary modes of operation--searching and browsing; (2) CIMARON2 terminal operator's guide is a step by step use of the system through an on-line computer terminal; (3) The BROWSER2 terminal operator's guide describes a program which is an independent routine used to scan currently stored index files, to save index terms temporarily, and to obtain hard copy of the displayed terms; and (4) A user's guide to file building. [Related documents are LI 003610, LI 003611, and LI 003645 through LI 003647.] (Author/SJ)

FINAL REPORT
Project No. 7-1083
Grant No. OEG-1-7-071083-5068

THE CIMARON SYSTEM: MODULAR PROGRAMS FOR THE ORGANIZATION AND SEARCH OF LARGE FILES

By
Arjun K. Aiyer

Institute of Library Research
University of California
Berkeley, California 94720

September 1971

The research reported herein was performed pursuant to a grant with the Office of Education, U.S. Department of Health, Education, and Welfare. Contractors undertaking such projects under Government sponsorship are encouraged to express freely their professional judgment in the conduct of the project. Points of view or opinions stated to not, therefore, necessarily represent official Office of Education position or policy.

U.S. DEPARTMENT OF
HEALTH, EDUCATION, AND WELFARE

Office of Education
Bureau of Research

TABLE OF CONTENTS

Page

1. GENERAL DESCRIPTION OF CIMARON SYSTEM. 1

1.1 General Retrieval from Large Files.............................. 1
1.2 Searching and Browsing.. 4
1.3 The Formulation of CIMARON Search Requests...................... 4
1.4 CIMARON Search Logic.. 7
1.5 CIMARON Retrieval Display....................................... 8
1.6 File Generation and Organization................................12

2. CIMARON2 TERMINAL OPERATOR'S GUIDE 17

2.1 Overview..17
2.1.1 CIMARON2 Structure and Commands..............................17
2.1.2 Use of the Sanders 720 Keyboards.............................20
2.2 Entering CIMARON2...20
2.2.1 Logging In...21
2.2.2 Selecting the Data-Base......................................21
2.3 Entering Search Requests..21
2.3.1 Search Keys and Attributes...................................22
2.3.2 The AD Attribute...23
2.3.3 Search Requests in the Form of Boolean Expressions...........23
2.4 Search Results..24
2.4.1 Diagnostics..24
2.4.2 The List Overflow Warning....................................25
2.4.3 Unusual Results..25
2.4.3.1 None...26
2.4.3.2 All..26
2.4.3.3 'All but'...26
2.4.4 Normal Results...27
2.5 Record Retrieval..28
2.5.1 The List Control Commands....................................28
2.5.2 Record Display Format..29
2.6 Exiting from CIMARON2...31
2.7 Disastrous Ends in CIMARON2.....................................32
2.8 CIMARON2 Messages and Code Tables...............................32

3. BROWSER2 TERMINAL OPERATOR'S GUIDE 41

3.1 Overview..41
3.2 BROWSER Commands..42
3.2.1 Select Data Base Index File..................................42
3.2.2 Select Portion of Index File to be Displayed.................42
3.2.3 Advance the Display..43
3.2.4 Save an Index File Entry.....................................44
3.2.5 Display the Save Area List...................................44
3.2.6 Remove a Term from Save Area List............................44
3.2.7 Print...44
3.2.8 Close or Exit..45
3.2.9 Display the Available Commands or File Names.................45

TABLE OF CONTENTS (cont.)

Page

4. USERS' GUIDE TO FILE BUILDING 47

4.1 Overview 47
4.2 Creation of the Bibliographic Master File 47
4.3 Extraction of the Index Information 51
4.3.1 Parameter Control 51
4.3.2 ZODIAC Control Tables 53
4.3.3 Processing Sequence 54
4.3.4 Program Constraints 54
4.4 Sequencing the Index Data 55
4.5 Creation of the Index Files 55

LIST OF FIGURES

Figure Title Page

1. Data Collected by CIMARON 3

2. CIMARON Search Request Syntax Defined in Backus-Naur Form Notation 6

3. CIMARON Display Record Example 10

4. CIMARON Commands Available During Record Display Phase 10

5. General CIMARON Commands 11

6. Phase of CIMARON Operation and Legitimate Transfer Paths 17

7. General CIMARON Commands 18

8. CIMARON Commands Available During Results Display Phase 19

9. CIMARON2 Messages 33

10. Equivalence Table of Graphic Representations and Internal Codes (Listed in EBCDIC Sequence 34

11. Current BROWSER Commands 42

12. BROWSER2 Display of Index Terms 43

13. Finder File Record Format 49

14. ZODIAC Run Setup 50

15. FILOR Run Setup 52

16. ZODIAC Internal Control Tables 53

FOREWORD

This report contains the results of the second phase (July, 1968 - June, 1970) of the File Organization Project, directed toward the development of a facility in which the many issues relating to the organization and search of bibliographic records in on-line computer environments could be studied. This work was supported by a grant (OEG-1-7-071083-5068) from the Bureau of Research of the Office of Education, U.S. Department of Health, Education, and Welfare and also by the University of California. The principal investigator was M.E. Maron, Professor of Librarianship and Associate Director, Institute of Library Research; the project director and project manager were, respectively, Ralph M. Shoffner and Allan J. Humphrey, Institute of Library Research.

This report is being issued as seven separate volumes:

- Shoffner, Ralph M., Jay L. Cunningham, and Allan J. Humphrey. The Organization and Search of Bibliographic Records in On-line Computer Systems: Project Summary.

- Shoffner, Ralph M. and Jay L. Cunningham, eds. The Organization and Search of Bibliographic Records: Component Studies.

- Aiyer, Arjun K. The CIMARON System: Modular Programs for the Organization and Search of Large Files.

- Silver, Steven S. INTX: Interactive Assembler Language Interpreter Users' Manual.

- Silver, Steven S. FMS: Users' Guide to the Format Manipulation System for Natural Language Documents.

- Silver, Steven S. and Joseph C. Meredith. DISCUS Interactive System Users' Manual.

- Smith, Stephen F. and William Harrelson. TMS: A Terminal Monitor System for Information Processing.

Because of the joint support provided by the Information Processing Laboratory Project (OEG-1-7-071085-4286) for the development of DISCUS and of TMS, the volumes concerned with these programs are included as part of the final report for both projects. Also, the CIMARON system (which was fully supported by the File Organization Project) has been incorporated into the Laboratory operation and therefore, in order to provide a balanced view of the total facility obtained, the volume is included as part of the Laboratory project report. (See Maron, M.E. and Don Sherman, et al. An Information Processing Laboratory for Education and Research in Library Science: Phase 2. Institute of Library Research, 1971.)

ACKNOWLEDGMENTS

The CIMARON system evolved both with the help of many members of the Institute staff and with its use by faculty and students of the School of Librarianship.

In particular, I would like to acknowledge the many fruitful discussions with Chakravarthi Ravi during the design of the on-line search and retrieval components. Ralph Shoffner and William Schieber laid out the broad specifications of the system, while Jay Cunningham and Allan Humphrey provided many helpful suggestions.

The BROWSER subsystem was programmed by William Harrelson. Steve Smith, William Harrelson and Rodney Randall provided excellent system support.

Special thanks are due to Ed Mignon and Don Sherman, not only for their introduction of the use of CIMARON to Library School students, but also for their observations and comments.

Principal acknowledgments are due to the School of Librarianship of the University of California and the Office of Education of the Department of Health, Education and Welfare, for making this work possible.

In addition, I wish to thank and to commend the work of the Institute personnel who prepared these pages for publication: notably Carole Fender, Linda Herold, Barbara Johnson, Jan Kumataka, Pat Oyama, and Rhozalyn Perkins.

1. GENERAL DESCRIPTION OF CIMARON SYSTEM

1.1 General Retrieval from Large Files

The File Organization Project* has made available a set of programs which are designed to operate on large files of bibliographic records, typically machine-form catalog entries for monographs. From the educational and research point of view, these programs are designed as an instrument for understanding and refining the techniques of bibliographic search. At present, access is provided to two data bases in the MARC II record structure:

a. 95,000 records, representing approximately 65% of the holdings of the library of the University of California at Santa Cruz. By 1971, this file will grow to 120,000 records and represent over 80% of the Santa Cruz campus holdings.

b. 5,700 records, representing a portion of the collection of the University Hospital, U.C., San Diego. This smaller file is focused almost entirely on medical topics.

The retrieval program, CIMARON, in common with other programs in the Information Processing Laboratory, operates interactively with the students and researchers who use it. The files are organized so that they can be searched "on-line," i.e., while the user waits. In most cases, searches are performed in less than ten seconds.

Any search request may consist of a series of search keys connected by Boolean operators and utilizing parentheses. Allowable search keys for a given data base are specified at the time the data base is locked into the system. For the San Diego file, four keys currently are allowable: Author, Subject, Title, and Dolbyized Author.** For the Santa Cruz file, due to present limitations of available disc space, only Author and Subject search keys are permitted. In principle, other search keys such as Series, Publisher, Publication Date, Class Number, Dewey Decimal No., etc., can be generated. The search key lists are as follows:

Current	Planned
AU/ - Author	SE/ - Series
TI/ - Title	PU/ - Publisher
SU/ - Subject	PD/ - Publication Date
AD/ - Dolbyized Author	CN/ - Class Number or Call Number
	DD/ - Dewey Decimal Number

In CIMARON, search requests consist of a set of Search Keys, having an explicit relationship between them. The user defines this relationship in terms of three Boolean connectives: AND, OR, NOT. The meanings of these connectives are as follows:

*USDHEW, Grant No. OEG-1-7-071083-5068.

**Dolbyized Author Names refer to a process of association names that are similar phonetically but spelled differently: Tschaikovskii, Tshaicovsky, Chaikowski, etc.

'AU/FREUD' AND 'SU/DREAMS'	(All books written by Freud <u>and</u> about dreams)
'AU/FREUD' OR 'SU/DREAMS'	(All books written by Freud, as well as all books about dreams including those written by Freud)
NOT 'AU/FREUD' AND 'SU/DREAMS'	(All books about dreams, except those written by Freud)
NOT 'AU/FREUD' AND 'AU/FREUD'	(The null set)
NOT 'AU/FREUD' OR 'AU/FREUD'	(The universal set)

Further, CIMARON allows parenthetic search requests to be formulated:

'SU/DREAMS' AND ('SU/FREUD' OR 'SU/JUNG')	(All books about Freud's or Jung's work on dreams)
('AU/FREUD' or 'AU/JUNG') AND ('SU/DREAMS' OR 'SU/HYSTERIA')	(All books written by Freud or Jung on either of the two subjects, dreams or hysteria)

With such a powerful variety of options available, CIMARON users are able to explore a number of manual and computer specific search strategies. They are able to formulate comparisons between various manual and automated methodologies related to search formulation and search expansion. In addition, the users gain many direct insights into the structure of machine-form bibliographic records, especially the relationship between the identification of bibliographic data elements and the formulation of search requests.

The CIMARON program was planned as the core program within an expanding system devoted to experimentation with organization and search of large files of bibliographic data. As a result, CIMARON was designed as a modular program with separate segments for:

a. selection of data base to be searched;

b. negotiation of the search request;

c. analysis of the search request and index file search;

d. report of the search results;

e. retrieval and display of the master records;

f. search iteration or termination.

Also, an internal data logging procedure has been developed to provide extensive information about both the behavior of the users of the system and the internal operation of the system itself.* Students may be interested in this logging feature since it is anticipated that the data gathered will be useful for many different types of analyses. The data logged includes the identification of the user, the search request made, the amount of time spent in searching the file, etc. (see Figure 1).

*The computer code for this procedure was developed but was not fully operational at the time of writing this report.

FIG. 1: DATA COLLECTED BY CIMARON

Field #	Field Name	Bytes	
1.	LRECLEN	2	Length of this log record in hex
2.	LTERMNO	2	Terminal number in EBCDIC
3.	LUNAME	4	Initials of user
4.	LPNAME	8	Name of program
5.	LDATE	4	Julian date in packed format
6.	LINTIME	4	Time at entry in packed format
7.	LSEQNO	2	Sequence number in packed format
8.	LFLAGS	2	Bit flags indicating up to 16 conditions
9.	LBCODE	2	Data-base code in EBCDIC
10.	LDIACNT	2	Diagnostics count in packed format
11.	LIXRCNT	2	No. of index records read in hex
12.	LADXCNT	2	No. of master addresses read from disk in hex
13.	LTRKCNT	2	No. of tracks read from disk in hex
14.	LSRCCNT	3	Number of records reported after search in packed format
15.	LREFCNT	3	Number of records retrieved by user in packed format
16.	LQRTIM	4	Time at query entry in packed format
17.	LSRCTIMI	4	Time at start of search in packed format
18.	LSRCTIM2	4	Time at end of search in packed format
19.	LRETTIM1	4	Time at start of retrieval in packed format
20.	LRETTIM2	4	Time at end of retrieval in packed format
21.	LOUTTIME	4	Time at exit in packed format
22.	LQRYCEN	2	Length of query in hex
23.	LQRTXT	var. up to 256	Text of the query

CIMARON is a highly sophisticated program in its design and operation. However, some expertise also is required of the user, since some of the program features are rudimentary as yet. For example, the user must know the format to use when submitting requests, since the present request negotiation phase of the program is limited only to testing the syntax of the request rather than otherwise assisting him in formulating his request.

This document is intended both to serve as an adequate introduction to new users of the system and to provide a general description of the system to those with interest in bibliographic storage and retrieval systems. Subsequent chapters of this volume contain users' guides for these programs which specify the exact format of commands, sequence of operations, etc. The remaining sections of this chapter will describe CIMARON and the file building programs at an intermediate level of detail.

1.2 Searching and Browsing

The CIMARON system may be conceptualized as having two distinct but complementary modes of operation: <u>searching</u> and <u>browsing</u>. The <u>search</u> mode has as its object the formulation of retrieval requests, the evaluation of requests against a master file of indexed records, and the retrieval of a relevant subset of records. The <u>browsing</u> mode has as its goal the examination of the CIMARON index files, the extraction of appropriate index file entries, and the ultimate utilization of index file entries as components (i.e. terms) of search requests.

The browsing mode is crucial in the CIMARON system because it helps the user maintain control of search operations. By browsing, the user can ascertain:

- legitimate non-empty terms for search requests
- variations in the representation of legitimate search request terms
- exact format (punctuation, abbreviations, spelling, etc.) for search request terms
- which terms will lead to overflow conditions in retrieval requests

Browsing thus can be considered a part of the analysis which precedes actual searching and which is necessary to avoid inaccurate, illegitimate, or inappropriate search request terms.

While the capability of browsing through index files was conceived as part of the original CIMARON design, it has not been implemented as part of the initial CIMARON code. However, this capability is provided through the use of Browser, an independent routine, originally coded to aid programmer debugging. The operation of Browser will not be discussed here, both because its operation is straightforward and because it represents a temporary implementation of the CIMARON system. Its operating instructions are provided following those of CIMARON2.

1.3 The Formulation of CIMARON Search Requests

Because of the hardware and software resources available in the Information Processing Laboratory, CIMARON is able to operate in real-time, to communicate bi-directionally with users, and to utilize the Sanders Cathode Ray Tube (CRT) terminal video screens to obtain the search specification and to format and display its retrieval results. We use the general term "interactive mode" to cover all these aspects of request formulation, real-time search, immediate display of retrieval results, and user-program communication. This "interactive mode" is distinguished from "batch mode" processing by the immediacy of the communication cycle. In CIMARON, the interactive mode is used both to select data base, search key file, search request formulation, format of retrieval display, and to control the viewing of the retrieved records.

CIMARON begins with a descriptive summary of its data bases and Search Key Files. The user is asked to select a data base (SD=San Diego, SC=Santa Cruz), and CIMARON then opens the appropriate files of master file records plus associated Search Key files.

CIMARON then asks the user for a search request. A search request has a precise syntactic definition which is given in Figure 2 in Backus-Naur Form (BNF) notation. The basic syntactic components of the search request are:

a. Name of Search Key File (i.e., AU, TI, SU, AD)

b. Specification of search key value (e.g., FREUD or PSYCHOANALYSIS)

c. Boolean connectives between search terms (e.g., AND, OR, NOT)

d. Punctuation [e.g., apostrophe ' or slash / or reverse slash \ or left paren (or right paren)].

Apostrophes are used as left and right brackets around a search term. A slash is used to separate the name of the Search Key file from the search key value. Reverse slashes are used to bracket hexadecimal values to be entered into the search key value. (This is used to specify symbols which cannot be entered legitimately via the terminal keyboard, such as subfield delimiters \FA\ or an apostrophe within the search key value.)

Examples of CIMARON search requests are:

'AU/FREUD, SIGMUND'	(All books authored or co-authored by Sigmund Freud)
'AU/FREUD, SIGMUND' OR 'SU/FREUD, SIGMUND'	(All books by or about Sigmund Freud)
'SU/PSYCHOTHERAPY' AND ('AU/FREUD' or 'AU/JUNG' OR 'AU/ADLER')	(All books dealing with Psychotherapy and written either by Freud, Jung, or Adler.)

Before executing a Search Request, CIMARON checks the syntactic validity of the request and then rejects it if: (1) there are an uneven number of apostrophes; (2) the Search Key file is incorrectly specified; or (3) an unknown Boolean operator is used. The user is notified of the type of error and is given the option of correcting the search request which will be checked again for validity.

FIG. 2: CIMARON SEARCH REQUEST SYNTAX
DEFINED IN BACKUS-NAUR FORM NOTATION

```
 1.  <Search request>  : : =  <boolex>

 2.          <boolex>  : : =  <term>|<boolex>OR<term>

 3.            <term>  : : =  <factor>|<term>AND<factor>

 4.          <factor>  : : =  <operand>|NOT operand>

 5.         <operand>  : : =  <search key>|(<boolex>)

 6.      <search key>  : : = '<attrib. code>|<string>'

 7.    <attrib. code>  : : =  AU|TI|SU|AD

 8.          <string>  : : =  <alphanum>|\<hex>\ <String><alphanum>|<string>\<hex>\

 9.        <alphanum>  : : =  any string of EBCDIC characters excluding
                              apostrophe and reverse slash*

10.             <hex>  : : =  any string of hexadecimal digits, comprised
                              of legitimate 2-character hexadecimal
                              numbers, e.g., FO.
```

	Terminal Types
OR	AU
AND	TI
NOT	SU
(	AD
)	EBCDIC characters
'	Hexadecimal digits
/	

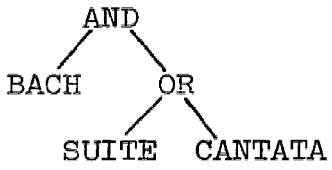
*Note that if an apostrophe, ', is to be included in the alphanumeric string, the hex representation for it must be provided. Otherwise it defines the end of the <search key>. Reverse slash, \, must be treated similarly.

The device used for user→program transmission is the terminal keyboard, with the CRT screen serving as a visual copy of what is being formulated and transmitted. The program→user communication is via the CRT screen. Because there are no mechanical linkages in this system (except for typing on the terminal keyboard), the interactive cycle is very rapid, usually on the order of less than two seconds. CIMARON may be in use at all three terminals at the same time, the small increase in delay time being due primarily to tying up the single telephonic link between the Laboratory and the Computer Center.

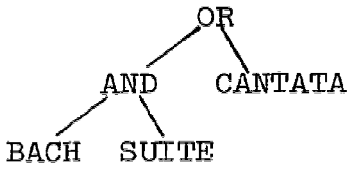
1.4 CIMARON Search Logic

When the user enters an acceptable search request, CIMARON begins operating according to search logic designed to maximize search efficiency. This attempt to maximize occurs at two levels. First, the search request is divided into components (corresponding to the terms of the search request), and the processing of these components is ordered so that the minimum evaluation of Boolean expressions is required. Second, the actual search operation is performed against the Access File (which is a sorted file), and Master File records are not examined at this stage of processing. The Access File contains the Search Key Values previously extracted by the File Generation subsystem and pointers to the master records. Thus, exhaustive search of the master records need not be performed by CIMARON.

The Boolean search request is treated by CIMARON as if it were an arithmetic expression; that is, there is a precedence order for evaluating expressions. That order is NOT, AND, OR. Further, all expressions within parentheses are evaluated before those without. Any search request thus can be treated as a simple binary tree. Each node of the tree is a Boolean operation with the tree leaves corresponding to the search terms in the search request expression. For example: 'AU/BACH' AND ('SU/SUITE' OR 'SU/CANTATA') can be represented as follows:



Note that without the parentheses, the expression 'AU/BACH' AND 'SU/SUITE' OR 'SU/CANTATA' would produce the following binary tree:



Thus the search logic can be seen in the following way. Beginning with the lowest level of the tree structure, a search is conducted for the term in the left leaf of the node. The search consists of both examining the appropriate Search Key Access* file and retrieving a list of addresses of Master File records which satisfy the Search Key value of the search request term. This list is called Left-List, and it is sorted into ascending order by Master File location address. A similar operation is performed for the right leaf of the node, and the resulting list of Master File addresses is sorted and stored in Right-List.

Once these two lists have been generated, they are combined into a third Result-List according to the Boolean logic specified at the node. If the node operator is OR, then Left-List and Right-List are additively combined, except that duplicate Master File addresses are combined into one entry. If the node operator is AND, then the Result-List consists only of Master File addresses that are present in both Left-List and Right-List (i.e., that are duplicated in both lists).

1.5 CIMARON Retrieval Display

Once CIMARON has completed execution of its search and retrieval logic and has created a final result-list of entries to be retrieved, it presents the requestor with a further set of options concerning the display of retrieved records. The user may specify briefly the number of records to retrieve and the format in which they should be displayed. After display of the records has begun, he may move forward or backward in the record display, ask for hard copy of a displayed record, or terminate the display. At that point he may either initiate another request or exit.

Four CIMARON messages are possible with regard to retrieval results:

a. NO RECORDS SATISFY REQUEST

b. XXX RECORDS SATISFY REQUEST - HOW MANY ARE DESIRED?

c. ALL EXCEPT XXX RECORDS SATISFY REQUEST--TYPE NONE OR XCEPT FOR THE EXCEPTIONS

d. CONGRATULATIONS - YOUR REQUEST SPECIFIES THE ENTIRE FILE

Message (a) indicates a zero result-list (no records match the search prescription and no records can be retrieved). The user is then allowed to exit or to reformulate his request by either altering the current request or submitting an entirely new request. The same options are offered to the user after message (d) which indicates a failure of a different type, namely that the search result encompasses the entire file (for example: 'AU BACH' OR NOT 'AU/BACH'). Message (b) is a more common result and gives the student an idea of how many records are potential candidates for retrieval and display. Message (c) is similar, but the class of retrieval records in this case is the negation (or exception to) the search request. For example, NOT 'AU/BACH', if taken literally, would result in specifying the entire file less those few titles authored by BACH. It is presumed that the student will work to display the exceptions, namely NOT (NOT 'AU/BACH'). The response to messages (b) and (c) can specify

*Recall that the Access File consists of Search Key values and linkages to the addresses of Master File records.

--- (null response is the same as ALL)

NONE (no records to be displayed)

ALL (all retrieval candidates to be displayed)

XXX (a three digit number, indicating how many records to display)

Any speficiation (except NONE) can be overridden during the display to terminate the display process.

When the user indicates the number of records to be displayed, the record display is initiated with a format default option (USER format) which presents all displayable data in the master record. This default option can be overridden at any time. Currently, two format options are available: (1) machine-form MARC II specified as MARC, and (2) user's MARC II specified as USER. The first option displays records in their original machine format; i.e., record leader, record directory, and data fields, in that order. This format is useful for students who are interested in analyzing the components of the machine-form MARC record. CIMARON is structured to accept other format display routines in the future.

The user format available is very similar to a normal catalog card, except that the major (MARC-defined) data fields each are printed on a separate line.* Each line begins with a short mnemonic identifying the contents of the line. These identifying codes are:

REC (accession number, publication date and call number)

MEH (main entry heading)

TIT (title)

IMP (imprint)

PAG (collation statement)

SER (series notes)

NOTE (other notes)

SUB (subject tracings)

OTH (author, title or series tracings)

A typical record under this format appears as Figure 3.

Records are displayed in disc address order and, therefore, are not alphabetized. At the end of each record display, the student has the following options: create a hard-copy version of the record currently displayed; to continue to the next record in the display sequence; skip forward or backward in the sequence; kill the display and return to the request formulation stage; or change display format.

*Accession number, publication date and call number all are printed on the first line, in that order.

FIG. 3: CIMARON DISPLAY RECORD EXAMPLE

```
REC:    0001234                 1965                 BL1032 B4
MEH:    BELLAH, ROBERT NEELLY, 1927
TIT:    RELIGION AND PROGRESS IN MODERN ASIA.  EDITED BY ROBERT N. BELLAH.
IMP:    NEW YORK, FREE PRESS, 1965
PAG:    XXV, 246 P., 22 CM.
NOTE:   REPORTS OF A CONFERENCE HELD IN MANILA IN 1963 UNDER THE AUSPICES
        OF THE CONGRESS FOR CULTURAL FREEDOM
SUB:    ASIA--RELIGION
SUB:    RELIGION AND SOCIOLOGY
SUB:    ECONOMIC DEVELOPMENT
CTH:    CONGRESS FOR CULTURAL FREEDOM
```

FIG. 4: CIMARON COMMANDS AVAILABLE DURING RECORD DISPLAY PHASE

----	(null response, display next record)
Kill	(go to request formulation/exit branch; return to the display can still be accomplished through command Bn)
Bn	(display the nth record back in the list; n=0, or n>current position provides the first record)
SKn	(skip n records and display record following; n=0, or n>remaining records provides the last record)
HARD	(send a copy of this record to the printer file, no change in display)
MARC	(display current and following records in MARC II communications format)
USER	(display current and following records in user format)

FIG. 5: GENERAL CIMARON COMMANDS

Commands	Program Phase				
	1	2	3	4	5
	Data Base Selection	Request Formulation	Results Display	End Results Display	File Closing
Finis	-	-	-	-	Terminate CIMARON
Exit	-	-	-	Go to Phase 5	-
Reopen	-	-	-	-	Go to Phase 1
SD	San Diego	-	-	-	-
SC	Santa Cruz	-	-	-	-
(null)	-	-	Show next record	Go to Phase 2	Terminate CIMARON

1.6 File Generation and Organization

Before CIMARON can be used, there must be a data base created on which it can operate. Thus, a sequence of programs is required to transform a random collection of machine-form bibliographic records into well-organized files which are easily searchable by CIMARON. These programs do not operate interactively nor do they provide real-time response capability. Rather, they operate in a batch-mode job stream, and the major successive steps are:

a. loading the basic Master File records into disc memory (FILOR)

b. extracting indicated Search Keys from each individual record (ZODIAC)

c. special search key generation (e.g., DOLBY)

d. sorting the collection of Search Keys (OS Sort)

e. consolidating Search Keys into a linked file structure (PAX)

The first step in this job stream is to construct on disc (at present, the IBM 2314) a linear array of the master file of input records. No ordering or pre-sorting of the input tape is required. The program reads in streams of variable length records, blocked or unblocked, and develops disc storage algorithms based upon the four-byte binary record descriptor word (RDW) in the standard IBM location. Possible input sources would be either magnetic tape or a sequential file on disc.

This program, called FILOR, creates two output files. The first is the Master File loaded sequentially onto the disc. These records are packed to the highest density possible; that is, they are blocked to the full 2314 track capacity of 7294 bytes. Because of this dense packing, some records may be split across two disc tracks. Although this increases programming complexities in the record display portion of CIMARON, the net savings in storage space is considerable due to the large size of bibliographic records.

The second file created by FILOR is a sequential list of the disc addresses of each Master File record. The data in each record of this Finder File consists of four elements:

a. Master File record number (sequentially assigned)

b. disc track number

c. relative position of the master file record within the disc track (this is termed the offset)

d. length of the master file record

There is one record in the Finder File for each Master File record. The data in the Finder File is used in subsequent programs to retrieve the records in the Master File.

The second step in the file organization process of CIMARON is to analyze each Master File record and to extract Search Keys from each record.

This process is performed by ZODIAC, which is organized to process MARC II structured records.* The definition of what constitutes a Search Key is controlled by parameter cards which are stated in terms of a set of MARC II major field tags. An example would be the generation of author (AU) Search Keys, consisting of:

100	Personal Name	Main Entry
110	Corporate	Main Entry
111	Conference	Main Entry
130	Uniform Title	Main Entry
700	Personal Name	Added Entry
710	Corporate	Added Entry
711	Conference	Added Entry
730	Uniform Title	Added Entry

The definition of the Search Key also could be narrowed, for example, by the exclusion of Corporate, Conference, and Uniform Title headings, or by the exclusion of Added Entries. Similarly, the definition could be expanded by adding Series Headings (400, 410, 411, 800, 810, 811) and/or Subject Tracings for Personal, Corporate, Conference, or Uniform Title names (600, 610, 611, 630).

The extraction of Search Keys occurs by using as input the two files generated in the previous pass; i.e., the disc-stored Master File and the Finder File. The Finder File record and its corresponding disc-stored Master File record are processed together. Each Master File record is analyzed for the existence of one or more Search Keys. For each Search Key found, a fixed-length output record is generated consisting of:

a. Search Key (all upper case)

b. Field Tag which caused the generation of Search Key

c. disc track number of Master File record

d. offset of record location within specified disc track

e. length of Master File record

A Master File record of course may have more than one Search Key, in which case multiple output records are generated.

ZODIAC will accept definitions of up to twelve Search Key files, and will, therefore, generate in a single pass Search Key files for author, title, subject, etc. Currently there are two constraints to the program. First, if several Search Key files are defined in a single ZODIAC program pass, the field tags which comprise a Search Key file must be unique and mutually exclusive. Thus, if one were creating two separate Search Key files, one for Author (AU) and one for Subject (SU), the 600 field (Personal Name Subject Tracing) could be allocated to one but not both of these files on a single ZODIAC run. To achieve placement in both, a second run of ZODIAC must be made with different parameter cards. The second constraint concerns the

*Library of Congress, Subscriber's Guide to the MARC Distribution Service, Washington, D.C.: Information Systems Office, 1970, p. 26.

ability to access subfields. In some situations, it would be useful to exclude certain subfields from the Search Key (e.g., $e relator), or to construct Search Keys from subfields altogether (e.g., a Key file consisting only of geographic subject heading subdivisions). Currently, the level of Search Key definition does not extend beyond the tagged field level.

Utilizing the output file of ZODIAC, other routines can be applied to generate "special" Search Keys. One such routine, DOLBY, provides the ability to generate Search Key files adapted to "noisy" or uncertain search specifications. The Search Key file, called Author-Dolby (AD), is obtained by scanning the output from ZODIAC and reducing the author surname to a canonical or quasi-phonetic representation. In this form all vowels are eliminated from the surname, and phonetically related consonants are reduced to single forms. This Search Key file (AD) may be used to produce a set of candidate retrievals when the user is uncertain of the pronunciation and/or orthography of an author search. A second "noisy" Search Key file generator has been programmed (but not yet implemented) for operation on the title field. The resulting Search Key file will contain a permuted sequence of information bearing content words, so that title requests can be processed effectively even where the searcher omits an article or preposition or confuses the order of the words in a title.

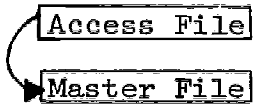
After the Search Key records have been generated, they are sorted using an IBM utility sort routine. The order of sort is: File Name (AU, SU, TI, etc.); Search Key, block/track number, offset. These sorted files are then presented to PAX, the routine which constructs the random access linked file structure. The standard access arrangement for indexed files, which is provided by IBM, has had the dual requirement that the Search Key be unique (no two records with the same key value) and that the records be the same length. Therefore, PAX analyzes the input records for duplicate key values and processes all those with the same value as a string. A Search Key string, therefore, is defined as one or more occurrences of the same value of the Search Key.

An example from a hypothetical AU file follows:

	Master File Record Address			
Search Key Value	Track	Offset	Length	
BACH, CHRISTIAN	51	1343	600	String 1
BACH, JOHANN SEBASTIAN	14	545	712	String 2
BACH, JOHANN SEBASTIAN	14	2506	455	
BACH, JOHANN SEBASTIAN	30	1025	512	
BACH, KARL PHILIPP EMANUEL	8	7210	650	String 3
BACH, KARL PHILIPP EMANUEL	13	150	475	
BACH, WILHELM FRIEDEMANN	47	4570	580	String 4

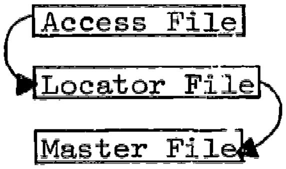
If a string consists of only one record, then the author is represented by only one title in the file. A string also may consist of many Search Key records, indicating multiple entries for an author in the Master File. In the example above, strings one and four are single-record strings, whereas strings two and three are multi-record.

Following this analysis, PAX creates two new files: a Key Access file, and a Key Locator file. There are two possible linkages from these files to the Master File, as follow:



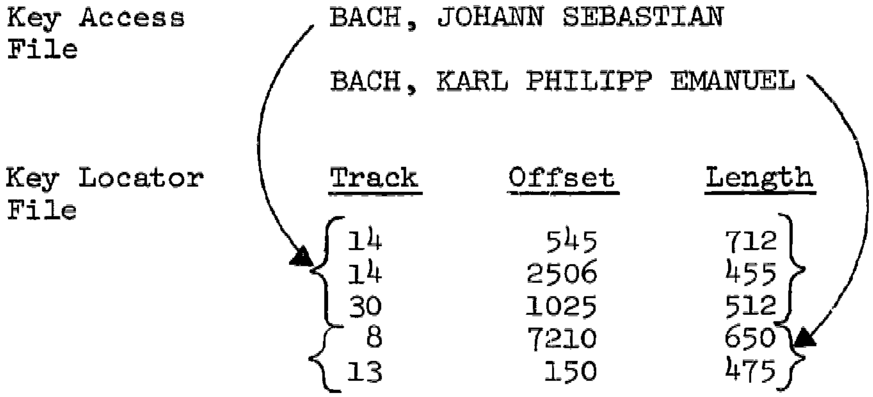
Used where only one master record has the key value.

or



Used where more than one master record has the same key value.

The simpler link occurs for each single-record string in the sorted Search Key file. In that case there is only one Master File Record in the data base corresponding to the Search Key value in the Key Access file record. Consequently, there is only one Master File record address and that may be carried directly in the Key Access record. The second case occurs for each multi-record string in the sorted Search Key file. In that case the Key Access file carries the Search Key value, but not the full set of Master File Record addresses. These addresses are stored instead in sequential fixed length records of the Key Locator file. The address in the Key Access file is thus a pointer to a sequential string of records in a second-level Locator file, each of which in turn points to a Master File record. Thus, to return to our example of multi-record strings:



Key Access File

BACH, JOHANN SEBASTIAN

BACH, KARL PHILIPP EMANUEL

Key Locator File

Track	Offset	Length
14	545	712
14	2506	455
30	1025	512
8	7210	650
13	150	475

Thus consolidation occurs as a result of carrying each unique Search Key value only once, thus reducing all multi-record strings to single-record strings. In order to maintain a fixed-length record structure in the Key Access file, Master File record addresses are transferred to a separate (Key Locator) file. This two or three level linked file structure is the final outcome of the file construction process and represents the data base utilized by the search logic and retrieval portions of the CIMARON program.

2. CIMARON2 TERMINAL OPERATOR'S GUIDE

2.1 Overview

This document is a terminal operator's guide to CIMARON2 - the latest version of an on-line search and retrieval facility, implemented at the Institute of Library Research, Berkeley. CIMARON2 allows the user to enter search requests (in a Boolean language) involving authors, titles, and subjects, from a remote terminal and subsequently presents the search results at the same terminal.

The hardware facility includes three SANDERS 720 character display terminals at the Information Processing Laboratory in the Library School and an IBM 360/40 computer with a 2314 disk storage unit at the Campus Computer Center. The software facility was developed under a Terminal Monitor System (TMS) designed specially for the Institute's on-line computing needs.

The guide is written with a view to conducting the reader through a complete session with CIMARON2 at the terminal. All text appearing on the display screen, whether generated internally by programs or keyed in by the user, is shown throughout this document in upper-case letters.

2.1.1 CIMARON2 Structure and Commands

In order to provide proper control and transfer between the program components, CIMARON and its command structure are divided into six phases. During each phase of the program, the user may choose from available options in order to transfer to other phases of the program, to submit a request, view the results, etc. Figure 6 lists these program phases and the legitimate transfers between them. Figure 7 shows the general commands available to the user during these phases. And finally Figure 8 shows the commands available during the results display phase to assist the user in viewing the retrieval result.

FIG. 6: PHASES OF CIMARON OPERATION AND LEGITIMATE TRANSFER PATHS

Phase #	Phase Type	Next Phase(s)
1	Data Base Selection	2
2	Request Formulation	2, 3, 6
3	End Search	2, 4, 6
4	Start Display	4, 5
5	End Display	2, 4, 6
6	File Closing	1, Terminate CIMARON

FIG. 7: GENERAL CIMARON COMMANDS

Program Phases ——→

Available Commands*	1 Data Base Selection	2 Request Formulation	3 End Search**	4 Start Results Display	5 End Results Display	6 File Closing
//CLOSE	-	Go to Phase 6	see CLOSE	-	see CLOSE	-
EXIT	-	-	-	-	-	Terminate CIMARON
REOPEN	-	-	-	-	-	Go to Phase 2
SD	San Diego	-	-	-	-	-
SC	Santa Cruz	-	-	-	-	-
(null)***	Santa Cruz	Search same request	Go to Phase 2	Show all records	Go to Phase 2	Terminate CIMARON
CLOSE	-	-	Go to Phase 6	-	Go to Phase 6	-
RESTART	-	-	Go to Phase 2	-	Go to Phase 2	-
EDIT	-	-	Go to Phase 2	-	Go to Phase 2	-

*Dash indicates command unavailable in this phase.

**End search is the phase arrived at if the search "failed"; Phase 4 is arrived at if the search "succeeded."

***(null) indicates depression of SEND BLOCK key with no typed response.

FIG. 8: CIMARON COMMANDS AVAILABLE DURING RESULTS DISPLAY PHASE

Available Commands*	Meaning
(null)	Display next record
KILL	Go to Phase 5, End Results Display; return to the display can still be accomplished through command Bn
BACK n	Display the nth record back in the list; n=0, or n>current position provides the first record; B and B1 are the same
SKIP n	Skip n records and display record following; n=0, or n>remaining records provides the last record
HARD	Send a copy of this record to the printer file, no change in display
MARC	Display current and following records in MARC II communications format
USER	Display current and following records in user format

*Underlined letters are the minimum typed characters to uniquely define the command. Additional characters are optional.

2.1.2 Use of the Sanders 720 Keyboards

The Sanders 720 keyboard consists of two major groups of keys - the alphanumeric group containing upper-case letters, digits, and punctuation characters and the function key group on the right hand side of the keyboard. Of the former group, the user is not concerned currently with the HOME, horizontal and vertical TAB and CR keys, because all formatting is usually controlled by the programs, and user input is expected to be a pure alphanumeric string. Of the latter group, the user is concerned with the INSERT, DELETE, and SEND BLOCK keys. The INSERT and DELETE (blue keys) in conjunction with the cursor positioning key (i.e., the SPACE key), enable the user to edit his input string before dispatching it with the SEND BLOCK function key.

Any message from within the program requires a user response. This may be a simple acknowledgment on his part that he has read what is on the screen, or it may be a command word or a search request. In any case the following can be noted:

a. The cursor is positioned one or two lines below the last line of the program message, often just beyond a right arrow (>) and blinks steadily.

b. The user types in his input here, edits it if necessary, and dispatches it by pressing the SEND BLOCK function key.

c. Both the message and the user's input are visible on the screen until the next message appears.

d. Sometimes, when all three terminals are being used, the user's response may go into a blink mode. This usually does not last more than a few seconds and means the input is being queued before it is processed.

Also, whenever CIMARON2 prompts the user to select one of many command options, the first alternative listed is usually a default option which may be taken merely by pressing the SEND BLOCK key. This causes a zero-length message to be sent to CIMARON2 as a signal for the default command. A simple SEND BLOCK (i.e., a zero-length message) is used elsewhere as an acknowledgment from the user. This is the same convention if viewed as a default command to "continue." This procedure allows the user to proceed rapidly along the most commonly used program paths.

2.2 Entering CIMARON2

CIMARON2 may be called up on the terminal as a user service under the Terminal Monitor System. This means that TMS brings into core a copy of the CIMARON2 machine code stored in a program library on disk and initializes the communication path between CIMARON2 and the appropriate terminal. Since the program is reentrant, only one copy of it is in core at any time, although communication paths may have been established from CIMARON2 to more than one terminal. The detailed procedure for entering CIMARON2 is described below.

2.2.1 Logging In

When TMS has been initiated, the following messages appear on each screen:

```
TMS100I - TMS IN OPERATION

TMS101A - WAITING FOR LOG IN
```

The normal response to this is to type in either GP01 or GP02 and send the message (depress the SEND BLOCK button). The initials of some ILR personnel are also valid, but these are usually entered by those persons when conducting debugging sessions or when running machine tutorial programs (i.e., DISCUS). If one of the valid initials is received (say GP01), TMS responds with:

```
TMS102I - GP01 LOGGED IN

TMS104A - SPECIFY PROGRAM
```

The terminal user has been logged in successfully and may now call CIMARON2 by typing CIMARON2 followed by a SEND BLOCK. This results in entry to the program and the display of a "title-page" message.

2.2.2 Selecting the Data-Base

The initial "title-page" message lists the version of the program, the date this version was first operational, the data bases currently stored and indexed on disk, and the attributes via which the data base master files can be searched. The last line requests the user to select the data base, which he does by typing a two letter code. Currently, there are two data-bases, and the codes are SC (for Santa Cruz University Catalog) and SD (for San Diego Biomed Catalog). The Santa Cruz data-base will be opened by default. The selection of the data-base at this point ensures that all subsequently opened index (attribute) files will pertain to the correct data-base. The code letters typed by the user will appear in the top left-hand corner of the screen (HOME position). If the code letters are invalid, the message SELDB will reappear, allowing the user to try again.

2.3 Entering Search Requests

Search requests are entered when the following prompt appears on the top line of the screen:

```
CIMARON IS READY - ENTER BOOLEAN EXPRESSION:
```

The cursor is positioned two lines below this message, and the user may type in the expression immediately. As always, user input is dispatched to the computer by a SEND BLOCK at the end of the input. If the input string typed by the user is detected to be empty (zero length), the above message (MSG1) reappears, allowing the user to try again.

Three other options are provided here. One is to allow the user to type//CLOSE, instead of a search request. Section 2.6 explains what happens when CIMARON2 receives this command. The second is to allow comments to be entered in the system log. A comment must begin with an asterisk and may be up to 256 characters long; comment strings longer than 256 characters will be truncated. The third option arises when //SPEC causes a program interrupt and transfers to the interrupt handler. This option is for the use of programmers only.

2.3.1 Search Keys and Attributes

The minimal form of an input request is a search key. A search key is enclosed in single quotes and the first two characters identify the type of attribute (associated with master file records) that the search key represents. E.g.,

'AU/KINSEY'

'TI/THE HISTORY OF MEDICINE'

These represent search keys of the author and title variety respectively. The attribute codes are AU and TI, and the slash following the code is a mandatory delimiter. Currently, the legal attribute codes are AU, TI, SU, AD. The last two represent subject and "Dolbyized" author. Note that every character between the slash and the ending quote is significant, including blanks, and is used in direct comparison with keys in index files. In actuality, an inclusive Boolean OR is performed between the associated master records of every key in the index file whose first part matches (character for character) the submitted search key. This will henceforth be referred to as the part-key facility. At present, all user-submitted search keys are part-keys because they are truncated to 32 characters, (including the attribute code and delimiter), before they are entered in tables, whereas the index files have key lengths of 40 or more.

It is important to realize, however, that in order to get a unique match on a single record (key) in the index file, it is necessary only to enter a sufficiently long part-key by consulting the index file listing.* An additional facility provided in this respect is defining hexadecimal strings within the search key. A hexadecimal string is identified by the fact that it is enclosed in reverse slashes. E.g.,

'SU/NURSING\FAA7\OBSTETRIC'

is a subject search key containing the hexadecimal string FAA7 (a MARC II subfield delimiter followed by a lower-case x). This device is used whenever characters to be included in a search key are not to be found on the terminal keyboard, and gives access to the entire 8-bit character set.

*Also known as authority listings. These are essentially printouts of the author, title, and subject index files showing the keys and the number of master records indexed under that key.

2.3.2 The AD Attribute

The letters AD represent the "Dolbyized" authors attribute. Search keys preceded by this code are routed to a special index file in which the keys are canonical forms of author names. The canonical form is obtained by applying a series of transformations (rules) to an author name. These rules, first proposed by Dolby,* are designed to smooth out differences due to minor spelling variations (or errors) in English surnames, and reduce the variants to a canonical form. Thus by applying the algorithm to a search key and then looking for hits in an index file of canonical names, one obtains noisy match, i.e., one achieves greater recall at the expense of precision.

The "Dolbyized" author index files contain not just author names (in canonical form) but also associate authors, editors, translators, etc. This further improves recall.

For example, searching the San Diego Biomedical file with the key 'AD/KINSEY' results in retrieving 10 records of which

2 were authored by KINSEY

1 was about KINSEY

4 were authored by KUNTZ

1 was authored by KOONTZ

1 was authored by KINSMAN

1 was authored by KUNSTADTER

In the last two names, only the first syllable is "close" to KINSEY. This is because the part-key feature was automatically invoked after the key typed by the user was "Dolbyized." In order to suppress the part-key search on the Dolby file, the user may add a blank to the end of the key thus: 'AD/KINSEYɓ'. In this case, the last two names, viz., KINSMAN and KUNSTADTER, no longer appear in the retrieved records.

2.3.3 Search Requests in the Form of Boolean Expressions

So far, search requests composed of a single search key have been described. However, CIMARON2 accepts more complex requests in the form of parenthesized Boolean expressions wherein search keys with different attributes can be mixed freely. For example, a more complex request would be:

```
'SU/OPTICS' AND ('TI/FIBER OPTICS IN SURGERY OF THE EYE'
OR 'AD/KOONS') AND NOT ('SU/LENSES' OR 'SU/TECHNOLOGY')
```

*Dolby, James L., "An Algorithm for Noisy Matches in Catalog Searching," in Cunningham, Jay L. et al., A Study of the Organization and Search of Bibliographic Holdings Records in On-Line Computer Systems: Phase I, Berkeley: Institute of Library Research, University of California, March 1969.

This expression has five search keys, three attribute types - SU, TI, and AD, and uses all three Boolean operators - AND, OR, NOT. The operators have an implied precedence as follows:

NOT	highest
AND	next highest
OR	lowest

The NOT is a unary operator and associated with the search key or sub-expression immediately to its right. The AND and OR are binary operators having a left-operand and a right-operand, each of which may be a search key or sub-expression. Between similar operators the implied precedence is left-to-right. Parentheses are used to define explicitly a group of operations of higher precedence: the higher the nesting level, the higher the precedence.

There are two important limitations on the complexity of the requests the user may type in:

a. The maximum number of search keys allowed in the expression is sixteen (16).

b. The maximum length of the search request, including all blanks, is 256 characters.

Since a line of the CRT screen accommodates 84 characters, long expressions have to be typed over more than one line. On typing beyond one line, the cursor automatically returns to the beginning of the next line, so no carriage control functions are required. On the other hand, by controlling the cursor position and using the INSERT and DELETE function keys, local editing of the serach request may be performed before dispatching it with a SEND BLOCK.

2.4 Search Results

There are many ways in which a search may end. It may end in a diagnostic due to an incorrect construction of the search request or due to storage overflow; it may result in reporting certain unusual results, or it may report that a finite set of records satisfies the user's request. These causes next are explained individually.

2.4.1 Diagnostics

CIMARON2 currently provides ten different diagnostics when something goes wrong either in the analysis of the input request or in the search. These are numbered DMØ through DM9. Of these, DMØ through DM6 report incorrect constructions, syntax errors, etc. in the input expression; DM7 and DM8 report file search failures; and DM9 is a special warning indicating partial search failure due to a storage block overflow being detected during the search. More will be said about DM9 in the next section. As an example, suppose the user entered the following search request:

```
'AU/SMITH' AND 'AU/JONES
```

with a missing ending apostrophe; CIMARON2 immediately responds with diagnostic message 3 (DM3) which appears at the top of the screen as follows:

```
UNBALANCED APOSTROPHES IN THE EXPRESSION - EDIT:
```

The improperly constructed expression is displayed 2 lines below this message. Once again, the user may employ the INSERT and DELETE function keys in conjunction with the cursor position control key (SPACE) to edit the message and send it. In the current example, adding an apostrophe after the S in JONES will correct the expression successfully and result in a search.

2.4.2 The List Overflow Warning

This is an interim feature in the program to detect and report the overflow of <u>any</u> of the internal record lists during the progress of the search. When corrective action is incorporated in CIMARON2 for this condition, the warning message (DM9) will be removed and the user will be unaware of the condition. An example of the appearance of DM9 is given below. An overflow condition is detected during the search for 'AU/TOYNBEE' in the Santa Cruz data base when the hundredth record indexed under this key is read, and DM9 appears thus:

```
*WARNING* - LIST OVERFLOW DETECTED, CUTOFF OCCURRED AT THE
                 FOLLOWING KEY:

AU/TOYNBEE, ARNOLD JAMES, D1819-

+ SEND BLOCK TO PROCEED +
```

The middle line shows the last key read from the author index file before overflow was detected; the last line indicates CIMARON2 is awaiting the user's acknowledgment. If the letters 'KKKK' appear just after the slash in the middle line, it means the last key read has no special significance since the overflow was detected during a Boolean OR operation.

When the user acknowledges this message by pressing SEND BLOCK, CIMARON2 proceeds to report the result of the search. The result will not reflect the true contents of the data base since some part of the search was prematurely ended. DM9 is the only diagnostic followed by a report of search results.

2.4.3 Unusual Results

Since search requests are in Boolean form and the concept of negation (using the NOT operator) is included, there are three types of "unusual" results that may occur. These are:

2.4.3.1 None

None of the records in the data-base file meet the conditions of the search request. This will occur in search requests specifying a conjunction of two mutually exclusive (record) sets, e.g.,

```
'AU/CHURCHILL, WINSTON' AND 'AU/HITLER, ADOLF'
```

CIMARON2 will report this search as follows:

```
NO RECORDS SATISFY REQUEST            (MSG2)

NO RETRIEVAL                          (MSG7)

OPTIONS ARE:  EDIT, RESTART, CLOSE    (MSG4)
```

The user is forced to bypass the record retrieval routines. The default response is EDIT, which results in the following message:

```
THE LAST SEARCH REQUEST WAS           (MSGE)
```

Below this is redisplayed an exact copy of the last search request which the user may edit using the function keys. If he wants to repeat the same search, he takes the default option by pressing SEND BLOCK. RESTART causes MSG1 to reappear on the screen and allows the user to submit a new search request. EXIT processing is described in Section 2.6.

2.4.3.2 All

All the records in the data base file are defined by the search request. This occurs whenever a search request specifies a disjunction of two complementary sets. For example,

```
'SU/HISTORY' OR NOT 'SU/HISTORY'
```

would result in the following messages:

```
CONGRATULATIONS-YOUR REQUEST SPECIFIES THE ENTIRE FILE    (MSG3)

NO RETRIEVAL                                              (MSG7)

OPTIONS ARE:  EDIT, RESTART, CLOSE                        (MSG4)
```

Again the user is not given the option of retrieving records.

2.4.3.3 'All but'

The last "unusual" case is one in which the search request specifies all but a small portion of the data base. This occurs for negated requests of the type:

```
e.g., 1   NOT   'AU/TOYNBEE, ARNOLD'

e.g., 2   NOT ('SU/PSYCHIATRY' OR 'SU/PSYCHOLOGY')
```

For these, the search results would be reported thus:

```
ALL EXCEPT XXX RECORDS SATISFY REQUEST
                                              (MSG6)
HOW MANY EXCEPTIONS ARE DESIRED?
```

Here, XXX represents a 3-digit number. The user, for obvious reasons, is permitted only to retrieve the exceptions. The allowable responses to this message are the same as those for MSG5 which appears after searches ending in normal results. (see Section 2.4.4)

2.4.4 Normal Results

In the usual case, the search ends by accumulating a finite number of records representing a small subset of the data-base. This is reported by CIMARON2 as follows:

```
XXX RECORDS SATISFY REQUEST
                                              (MSG5)
HOW MANY ARE DESIRED?
```

Once again, XXX stands for a 3-digit number denoting the search count. There are four allowable responses to this message and MSG6 (Section 2.4.3.3), namely: ALL, NONE, a number, or //FRMT.

ALL is the default and is taken to mean that the user wishes to look at all the records reported in the search count. It results in entry to the record retrieval routines (see Section 2.5).

NONE means the user wishes to bypass the retrieval routines. It results in the following messages:

```
END OF RETRIEVAL
                                              (MSG8)
ØØØ RECORDS DISPLAYED

OPTIONS ARE EDIT, RESTART, CLOSE, BACKUP      (MSG4)
```

BACKUP is one of the list control commands, and its use is explained in Section 2.5.1.

The user also may type a number less than or equal to the search count reported in MSG5 or MSG6. Appropriate validity checking is performed and a number greater than the search count is taken to mean ALL whereas any number evaluating to zero is equivalent to NONE.

FRMT is a command word indicating the user's desire to design the record display format before retrieval. Currently, the user is limited to a choice between two fixed formats. The following message appears when the user types //FRMT in response to MSG5 or MSG6:

```
SELECT RECORD DISPLAY FORMAT:  'MARC2' OR 'USER' (MSG9)
```

On typing either (MARC2 is the default) MSG5 or MSG6 (as the case may be) reappears, and either the number of records to be retrieved may be indicated, or the format may be changed once again.

If, after checking for the format command, CIMARON2 fails to find ALL, NONE or a number in response to MSG5 or MSG6, the following prompt appears:

```
YOUR OPTIONS HERE ARE:  ALL, NONE OR A NUMBER    (DMN)
```

Note that all the cases described above in Sections 2.4.3 and 2.4.4 may be preceded by the warning diagnostic (DM9), described in 2.4.2 in which case the reported search results are only partially right.

2.5 Record Retrieval

Several controls are provided to the user over the display of master file records. The first is selection of the number of records to be displayed. This is indicated either by responding ALL (the default response) to MSG5 or MSG6, or by typing a number less than or equal to the count reported by CIMARON2 in MSG5 or MSG6. The ALL response results in sequential retrieval of every master file record addressed by the result list (negated or not), in ascending order of disk address. Search results normally are displayed in this order unless the list control commands are employed to force a different sequence.

2.5.1 The List Control Commands

The List Control Commands provide the user with the ability to move freely forward or backward along the list of master records to be displayed, to change the display format between records, and to terminate retrieval after any record. At the end of any record display the allowable commands are:

K (for Kill) to stop further display of records

B (for Backup) to display the record previous to the current one

SK (for Skip) to display the record after the next one

M (for Marc) to redisplay the current record in MARC format

U (for User) to redisplay the current record in user format

H (for Hard-copy) to obtain a hard-copy of the current record.

Backup and Skip may be followed optionally by a number. Backup moves back 1 and displays that record, the Skip moves forward 1 and displays the record following it; thus B n (where n is a number of 1 or more digits) moves back n records, whereas SK n moves forward n records and displays the n+1 records. Also,

B 1   is equivalent to B

SK 1   is equivalent to SK

B Ø   always goes to the first record

SK Ø	always goes past the last record
B n	where n exceeds the search count is like B Ø
SK n	where n exceeds the search count is like SK Ø

M and U redisplay the current record only if the current format is U and M respectively, or else they just continue to the next record. In other words, if the format is changed, the current record is redisplayed in the new format; otherwise the next record is displayed in the same format. The hard-copy command sends the current display to the print file and makes no other change. Thus the user may type any of the other commands or just SEND BLOCK for the next record.

2.5.2 Record Display Format

The user has a choice of one of two display formats for each record. The two formats available are the LC MARC II format* and a more readable user format. The minor changes applied to the MARC II format prior to display are:

a. All codes other than those for punctuation and alphanumerics are translated to blanks. Such codes will be known as non-printing characters. The exceptions are as follows:

Original char. (hex. code)	Translated char.	Interpretation
1F	%	begin subfield (old MARC format)
26	+	end of field
37	*	end of record
FA	$	begin subfield

b. Lower-case alphabetics are translated to upper-case since the display terminals have no provision for these.

c. The following four punctuation characters are translated to blanks since they cause carriage control effects on the display screens:

Char.	Hex. code	Car. Control Effect
¢	4A	Horizontal tab
|	4F	Vertical tab
¬	5F	Carriage return
#	7B	Home cursor

*U.S. Library of Congress, Books: A MARC Format. Specifications for Magnetic Tapes Containing Monographic Catalog Records in the MARC II Format, 4th ed., Washington, D.C.: Information Systems Office, April 1970, 70 p.

The user format is a line-indented format with non-printing characters appearing as dots. The end of each field in the record is denoted by +, except for the last field which ends with * denoting end of record. The various fields are identified by brief mnemonics which are explained below:

Mnemonic	Type of field
REC	ILR Record accession number, published date and call number
MEH	Main entry heading (usually author)
TIT	Title
IMP	Imprint
PAG	Pagination
SER	Series note
NOTE	General notes
SUB	Subject heading
OTH	Other headings (usually co-authors)

In both formats, after displaying a record (or screenful, if the record is long enough) an acknowledgment is awaited from the user before the next record (or screenful) is displayed. This usually is requested below the last line of the record as:

```
+ SEND BLOCK TO PROCEED +   or

+ SEND BLOCK FOR NEW PAGE +
```

The user's usual response to these messages is to depress the SEND BLOCK key, whereupon a zero-length message, indicating that the user would like to proceed, is sent from the terminal. The user may, however, type a list control command at the end of a record (this may be confirmed by an asterisk at the end of the last field of the record). The command is typed just after a right arrow > at the beginning of the last line on the screen and is, naturally, followed by a SEND BLOCK. Note that on receiving a zero-length message (i.e., a pure SEND BLOCK) after any record, CIMARON2 assumes the user still wishes to retrieve a number of records he originally indicated in answer to MSG5 or MSG6, in the currently prevailing format until he types 'K', 'M', or 'U' after some succeeding record.

The retrieval process is ended in one of three ways: the user typed NONE or Ø to MSG5 or MSG6, the user typed 'K' after some record, or all the records requested by the user have been displayed. In either case, MSG8 followed by MSG4 appears thus:

```
END OF RETRIEVAL
                                            }     (MSG8)
XXX RECORDS DISPLAYED

OPTIONS ARE:  EDIT, RESTART, CLOSE, BACKUP        (modified MSG4)
```

The BACKUP option provided gives the user one last chance to get back into the display list. If this opportunity is not taken, the list is destroyed and can be recreated only with another search. As before, EDIT is the default option.

The action of CIMARON2 on receiving EDIT and RESTART has been indicated in previous sections. EDIT processing forms the search of the next section.

2.6 Exiting from CIMARON2

When the user types CLOSE in response to either MSG1 or MSG4, a delay of a few seconds occurs while CIMARON2 proceeds to close all the files either explicitly or implicitly opened by the user thus far. The data base was selected explicitly and opened by the user (see Section 2.2.2), whereas index files were implicitly opened the first time the associated attribute code was employed in a search expression. At the end of this "shut down" procedure, MSGØ appears:

```
ALL FILES CLOSED

OPTIONS ARE:  EXIT, REOPEN                        (MSGØ)
```

REOPEN indicates that the user wishes to begin search operations anew on a different data base (it would be wasteful to close a data base and reopen it immediately; therefore, EDIT or RESTART should be used to continue operations on the current data base), and results in the reappearance of the "title page" message described in Section 2.2.2. The user then may proceed as before with the new data base.

EXIT, which is the default response here, indicates that the user both is finished with CIMARON2 and will return control to the Terminal Monitor System, at which point the following TMS messages appear:

```
TMS106I - NORMAL EXIT FROM USER PROGRAM

TMS104A - SPECIFY PROGRAM
```

The user may at this point recall CIMARON2, but once again it is a wasteful exercise. If the user desires to sign-off and leave the terminal, he responds with LOGOUT, whereupon TMS comes back with:

```
TMS105I - XXXX LOGGED OUT

TMS101A - WAITING FOR LOGIN
```

where XXXX may be GPØ1, GPØ2, GPØ3, or the initials of some ILR personnel.

2.7 Disastrous Ends in CIMARON2

These are of two types:

a. Disasters ending in failure of TMS due to failure of the IBM operating system or hardware failure in the CPU or the teleprocessing system or due to one of the user programs damaging parts of TMS. Such conditions will cause abrupt loss of response from the terminal and require a reinitialization of TMS.

b. Disasters detected and trapped by TMS, originating from within CIMARON2. In such cases, TMS puts out appropriate messages about the nature of the disaster and purges that user's storage blocks, file buffers, and working areas. The user will have lost communication with CIMARON2; however, he can recall the program and begin anew. A typical example is given below:

Failure in opening a file. TMS puts out:

```
TMS1531 - ATTEMPT TO OPEN AN UNAVAILABLE/UNCATALOGED DATA SET

TMS110I - ABNORMAL RETURN FROM USER PROGRAM VIA PURGE ROUTINE

TMS104A - SPECIFY PROGRAM
```

For other such TMS messages, the user is referred to Appendix 2 in Part I of the TMS Users' Manual.*

2.8 CIMARON2 Messages and Code Tables

CIMARON2 ordinary messages are numbered MSGØ through MSG9; diagnostic messages are numbered DMØ through DM9. SELDB is the last line of the "title page" message. ACK is an acknowledgment request. The first line of multi-line messages usually appears at the top left-hand corner of the screen, i.e., the HOME position, while succeeding lines appear double-spaced below it. The messages currently available are listed in Figure 9. In the event that one wishes to interpret the internal codes for displayed or non-displayable characters, Figure 10 provides the equivalent codes. EBCDIC is the standard internal code in present files.

*Smith, Stephen F. and William Harrelson, TMS: A Terminal Monitor System for Information Processing, Berkeley: Institute of Library Research, University of California, 1971, p. 43-46.

FIG. 9: CIMARON2 MESSAGES

Message #	Text
MSGØ	ALL FILES CLOSED OPTIONS ARE: EXIT, REOPEN
MSG1	CIMARON IS READY - ENTER BOOLEAN EXPRESSION:
MSG2	NO RECORDS SATISFY REQUEST
MSG3	CONGRATULATIONS - YOUR REQUEST SPECIFIES THE ENTIRE FILE
MSG4	OPTIONS ARE: EDIT, RESTART, CLOSE
MSG5	XXX RECORDS SATISFY REQUEST HOW MANY ARE DESIRED?
MSG6	ALL EXCEPT XXX RECORDS SATISFY REQUEST HOW MANY EXCEPTIONS ARE DESIRED?
MSG7	NO RETRIEVAL
MSG8	END OF RETRIEVAL XXX RECORDS DISPLAYED
MSG9	SELECT RECORD DISPLAY FORMAT: MARC2 OR USER
MSGE	THE LAST SEARCH REQUEST WAS:
DMØ	THE EXPRESSION CONTAINS ADJACENT OPERATORS - EDIT:
DM1	INCORRECT USE OF THE "NOT" OPERATOR - EDIT:
DM2	UNBALANCED APOSTROPHES IN THE EXPRESSION - EDIT:
DM3	INVALID SYNTAX IN THE EXPRESSION - EDIT:
DM4	THE EXPRESSION CONTAINS ADJACENT OPERANDS - EDIT:
DM5	NO SEARCH KEYS IN THE EXPRESSION - RETYPE OR EDIT:
DM6	UNBALANCED PARENTHESES IN THE EXPRESSION - EDIT:
DM7	I/O ERROR IN SEARCHING INDEX FILE " " - EDIT:
DM8	ILLEGAL INDEX CODE " " IN SEARCH KEY - EDIT:
DM9	* WARNING * - LIST OVERFLOW DETECTED, CUT OFF OCCURRED AT THE FOLLOWING KEY:
SELDB	+ SELECT DATA BASE: TYPE "SD" OR "SC", THEN SEND BLOCK +
ACK	+ SEND BLOCK TO PROCEED +
DMN	YOUR OPTIONS HERE ARE: ALL, NONE OR A NUMBER

FIG. 10: EQUIVALENCE TABLE OF GRAPHIC REPRESENTATIONS AND INTERNAL CODES (LISTED IN EBCDIC SEQUENCE)

NAME				ASCII 6-BIT OCTAL		
	GRAPHIC	EBCDIC HEX	ASCII 8-BIT HEX	ESCAPE CODE*	CODE	
Null		ØØ	ØØ	73_8	ØØ	
		Ø1				
Double Underscore	‗	Ø2	F5	75_8	65	
Angstrom	°	Ø3	EA	75_8	52	
		Ø4				
		Ø5				
		Ø6				
Delete		Ø7	7F		77	
Circumflex	^	Ø8	E3	75_8	43	
Cedilla	¸	Ø9	FO	75_8	60	
Superior Dot	•	ØA	E7	75_8	47	
Left Hook	ȷ	ØB	F7	75_8	67	
Right Hook	ɩ	ØC	F1	75_8	61	
Inverted Cedilla	ʅ	ØD	F8	75_8	7Ø	
Hacek	ˇ	ØE	E9	75_8	51	
Acute	´	ØF	E2	75_8	42	
Double Acute	˝	1Ø	EE	75_8	56	
Umlaut	¨	11	E8	75_8	5Ø	
Dieresis	¨	12	FC	75_8	74	
Tape Mark		13	17		27	
		14				
		15				
Backspace		16	Ø8	73_8	1Ø	
Idle		17	16			
Candrabindu	ँ	18	EF	75_8	57	
Macron	¯	19	E5	75_8	45	
		1A				
Double Dot Below	..	1B	F3	75_8	63	
Dot Below	.	1C	F2	75_8	62	
Circle Below	°	1D	F4	75_8	64	
High Comma	'	1E	FE	75_8	76	
High Comma (off center)	'	1F	ED	75_8	55	
		2Ø				
		21				
		22				

*ASCII 6-Bit Code containing no escape code is in standard set.
Escape code =
73_8 - Non Standard Set I
75_8 - Non Standard Set II

FIG. 10 (Cont.)

NAME	GRAPHIC	EBCDIC HEX	ASCII 8-BIT HEX	ESCAPE CODE	ASCII 6-BIT OCTAL	
					CODE	
High Question	?	23	EØ	75_8	40	
		24				
Line Feed		25	ØA	73_8	12	
End of Field		26	1E	73_8	36	
		27				
Upadhmaniya	⌣	28	F9	75_8	71	
Tilde	~	29	E4	75_8	44	
		2A				
Grave	ˋ	2B	E1	75_8	41	
Breve	˘	2C	E6	75_8	46	
Double Tilde 1st Half	⌒	2D	FA	75_8	72	
Double Tilde 2nd Half	⌣	2E	FB	75_8	73	
		2F				
Ligature 1st Half	◜	3Ø	EB	75_8	52	
Ligature 2nd Half	◝	31	EC	75_8	53	
		32				
		33				
		34				
		35				
		36				
End of Transmission		37	1D	73_8	35	
		38				
		39				
		3A				
		3B				
Patent	®	3C	AA	75_8	12	
Flat	♭	3D	A9	75_8	11	
Open Bracket	[	3E	5B	73_8	73	
Close Bracket	]	3F	5D	73_8	75	
Space		4Ø	2Ø		ØØ	
		41				
		42				
		43				
		44				
		45				
		46				
		47				
		48				
		49				
		4A				
Period	.	4B	2D		16	
Less Than	<	4C	3C		34	

FIG. 10 (Cont.)

NAME	GRAPHIC	EBCDIC HEX	ASCII 8-BIT HEX	ESCAPE CODE	ASCII 6-BIT OCTAL CODE	
Open Paren	(	4D	28		1Ø	
Plus	+	4E	2B		13	
		4F				
Ampersand	&	5Ø	26		Ø6	
Mi͡agkiĭ Ƶnak	ʹ	51	A7	75_8	Ø7	
Tvërdyi Ƶnak	ʺ	52	B7	75_8	27	
Alif	ʼ	53	AE	75_8	16	
Ain	ʻ	54	BØ	75_8	2Ø	
		55				
		56				
		57				
		58				
		59				
Exclamation Point	!	5A	21		Ø1	
Dollar Sign	$	5B	24		Ø4	
Asterisk	*	5C	2A		12	
Close Paren	)	5D	29		11	
Semi Colon	;	5E	3B		33	
		5F				
Minus, Hypen	-	6Ø	2D		15	
Slash	/	61	2F		17	
		62				
		63				
Middle Dot	·	64	A8	75_8	1Ø	
		65				
		66				
		67				
		68				
		69				
British Pound	£	6A	B9	75_8	31	
Comma	,	6B	2C		14	
Percent	%	6C	25		Ø5	
Underline	_	6D	F6	75_8	66	
Greater Than	>	6E	3E		36	
Question Mark	?	6F	3F		37	
		7Ø				
		71				
		72				
		73				
		74				
		75				
		76				
		77				

FIG. 10 (Cont.)

NAME	GRAPHIC	EBCDIC HEX	ASCII 8-BIT HEX	ASCII 6-BIT OCTAL ESCAPE CODE	CODE	
		78				
		79				
Colon	:	7A	3A		32	
Cross Hatch	#	7B	23		Ø3	
At Sign	@	7C	4Ø	73_8	4Ø	
Prime, Apostrophe, Quote	'	7D	27		Ø7	
Equal	=	7E	3D		35	
Double Quote	"	7F	22		Ø2	
		8Ø				
Lower Case A	a	81	61		41	
Lower Case B	b	82	62		42	
Lower Case C	c	83	63		43	
Lower Case D	d	84	64		44	
Lower Case E	e	85	65		45	
Lower Case F	f	86	66		46	
Lower Case G	g	87	67		47	
Lower Case H	h	88	68		48	
Lower Case I	i	89	69		51	
Lower Case Æ	æ	8A	B5	75_8	25	
Lower Case Cross D	đ	8B	B3	75_8	23	
Lower Case Eth	ð	8C	B4	75_8	32	
Lower Case I (without dot)	ı	8D	B8	75_8	3Ø	
Lower Case Polish Ł	ł	8E	B1	75_8	21	
Lower Case Œ	œ	8F		75_8	26	
		9Ø				
Lower Case J	j	91	6A		52	
Lower Case K	k	92	6B		53	
Lower Case L	l	93	6C		54	
Lower Case M	m	94	6D		55	
Lower Case N	n	95	6E		56	
Lower Case O	o	96	6F		57	
Lower Case P	p	97	7Ø		6Ø	
Lower Case Q	q	98	71		61	
Lower Case R	r	99	72		62	
Lower Case Hook O	ơ	9A	BC	75_8	34	
Lower Case Slash O	ø	9B	B2	75_8	22	
Lower Case Thorn	þ	9C	B4	75_8	24	
Lower Case Hook U	ư	9D	BD	75_8	35	
		9E				
		9F				
		AØ				
		A1				

FIG. 10 (Cont.)

NAME				ASCII 6-BIT OCTAL		
	GRAPHIC	EBDIC HEX	ASCII 8-BIT HEX	ESCAPE CODE	CODE	
Lower Case S	s	A2	73		63	
Lower Case T	t	A3	74		64	
Lower Case U	u	A4	75		65	
Lower Case V	v	A5	76		66	
Lower Case W	w	A6	77		67	
Lower Case X	x	A7	78		7Ø	
Lower Case Y	y	A8	79		71	
Lower Case Ƶ	ƶ	A9	7A		72	
		AA				
		AB				
		AC				
		AD				
		AE				
		AF				
		BØ				
		B1				
		B2				
		B3				
		B4				
		B5				
		B6				
		B7				
		B8				
		B9				
		BA				
		BB				
		BC				
		BD				
		BE				
		BF				
		CO				
Upper Case A	A	C1	41	73_8	41	
Upper Case B	B	C2	42	73_8	42	
Upper Case C	C	C3	43	73_8	43	
Upper Case D	D	C4	44	73_8	44	
Upper Case E	E	C5	45	73_8	45	
Upper Case F	F	C6	46	73_8	46	
Upper Case G	G	C7	47	73_8	47	
Upper Case H	H	C8	48	73_8	5Ø	
Upper Case I	I	C9	49	73_8	51	
Upper Case Æ	Æ	CA	A5	75_8	Ø5	

FIG. 10 (Cont.)

NAME	GRAPHIC	EBCDIC HEX	ASCII 8-BIT HEX	ESCAPE CODE	ASCII 6-BIT OCTAL CODE	
Upper Case Cross D	Đ	CB	A3	75_8	Ø3	
		CC				
		CD				
Upper Case Polish L	Ł	CE	A1	75_8	Ø1	
Upper Case Œ	Œ	CF	A6	75_8	Ø6	
		DØ				
Upper Case J	J	D1	4A	73_8	52	
Upper Case K	K	D2	4B	73_8	53	
Upper Case L	L	D3	4C	73_8	54	
Upper Case M	M	D4	4D	73_8	55	
Upper Case N	N	D5	4E	73_8	56	
Upper Case O	O	D6	4F	73_8	57	
Upper Case P	P	D7	5Ø	73_8	60	
Upper Case Q	Q	D8	51	73_8	61	
Upper Case R	R	D9	52	73_8	62	
Upper Case Hook O	Ơ	DA	AC	75_8	14	
Upper Case Slash O	Ø	DB	A2	75_8	Ø2	
Upper Case Thorn	Þ	DC	A4	75_8	Ø4	
Upper Case Hook U	Ư	DD	AD	75_8	15	
		DE				
		DF				
		EØ				
		E1				
Upper Case S	S	E2	53	73_8	63	
Upper Case T	T	E3	54	73_8	64	
Upper Case U	U	E4	55	73_8	65	
Upper Case V	V	E5	56	73_8	66	
Upper Case W	W	E6	57	73_8	67	
Upper Case X	X	E7	58	73_8	7Ø	
Upper Case Y	Y	E8	59	73_8	71	
Upper Case Ƶ	Ƶ	E9	5A	73_8	72	
		EA				
		EB				
		EC				
		ED				
		EE				
		EF				
Zero	Ø	FØ	3Ø		2Ø	
One	1	F1	31		21	
Two	2	F2	32		22	

FIG. 10 (Cont.)

NAME	GRAPHIC	EBCDIC HEX	ASCII 8-BIT HEX	ESCAPE CODE	ASCII 6-BIT OCTAL	
					CODE	
Three	3	F3	33		23	
Four	4	F4	34		24	
Five	5	F5	35		25	
Six	6	F6	36		26	
Seven	7	F7	37		27	
Eight	8	F8	38		30	
Nine	9	F9	39		31	
Double Dagger	‡	FA	1F	73_8	37	
		FB				
		FC				
		FD				
		FE				
		FF				

3. BROWSER2 TERMINAL OPERATOR'S GUIDE

3.1 Overview

BROWSER2 is an independent routine, operating under TMS, which may be used to scan currently stored index files, to save index terms temporarily, and to obtain hard copy of the displayed terms. Our current development plans call for increasing the ease of communication between BROWSER and CIMARON, and eventually for their incorporation into a single program system with two operating modes.

Currently, however, BROWSER is a separate program and must be entered using BROWSER2 as the program name following the TMS program select command: TMS104A - SPECIFY PROGRAM. Once BROWSER2 is entered, two informational displays are available if positive replies are given to the first two BROWSER2 questions.

Q01: DO YOU WANT OPERATING INSTRUCTIONS?

A YES response results in a summary page of operating instructions being displayed (see Figure 11).

Q02: DO YOU WANT A LIST OF ACCESSIBLE FILES?

A YES response will display the current index file inventory.

The files currently available in this inventory are:

SCAU1 -	Santa Cruz	Author
SCSU1 -	Santa Cruz	Subject
SDAU1 -	San Diego	Author
SDTI1 -	San Diego	Title
SDSU1 -	San Diego	Subject
SDAD1 -	San Diego	Dolbyized Author

After the initial BROWSER informational screen displays, the major BROWSER command functions are:

a. Select a data base index file

b. Select which portion of the index file is to be examined

c. Advance the display

d. Save an index file entry

e. Display the Save Area List

f. Remove a term from the Save Area List

g. Print a hard copy version of the index file display or of the Save Area List

h. Exit from file examination or from BROWSER2

i. Display the available commands or file names.

FIG. 11: CURRENT BROWSER COMMANDS

COMMAND	MEANING OF COMMAND
sendblock	display advance
F xbl	forward x terms (x should be less than 100)
G 'xxxx'	get index entry xxx
s xbl	$1 \le x \le 10$ move line x to save area
R xbl	delete line x from save area
D	transfer from current display $\begin{bmatrix}\text{save area}\\ \text{index area}\end{bmatrix}$ to $\begin{bmatrix}\text{index}\\ \text{save}\end{bmatrix}$
H	hard copy output of current display
//Close	close current file
//Exit	exit from BROWSER

3.2 BROWSER Commands

3.2.1 Select Data Base Index File

Following the informational displays or on exit from a prior file, BROWSER2 requests that the user SPECIFY FILE NAME. The proper response to this is enter the name of any legitimate index file e.g. SCSU1. The index file name may be entered only as a response to the BROWSER2 request to specify a file name. Entering the name of a non-existent file will cause a syntax error, and BROWSER2 will request that the name be re-specified.

3.2.2 Select Portion of Index File to be Displayed

This command enables the user to specify an alphabetic key as an initial display value. The BROWSER2 message is:

SPECIFY KEY OR PARTIAL KEY.

The format of the response is:

G 'Key Value' (e.g. G 'LIBRARIES').

The attribute value (AU, SU, etc.) need not be specified, since it is implicit in the index file name which has been used for selection. BROWSER will use the alphabetic value of the Key Value entered to begin an alphabetically ordered display of index file entries. This is shown in Figure 12. The display contains ten entries, and each entry is numbered. The display also gives a count field which expresses the number of master file records which are linked to this index file entry. This count is effectively the number of works indexed by an individual index descriptor. If the G 'Key Value' gives a value which does not exist in the index file, then the display begins with the next legitimate index file entry.

FIG. 12: BROWSER2 DISPLAY OF INDEX TERMS

```
                                                  COUNT

 1.  LIBRARIES $X AFRICA $X DIRECT*               0001
 2.  LIBRARIES $X ANECDOTES, FACETIAE, SATIRE*    0001
 3.  LIBRARIES $X AUSTRALIA*                      0003
 4.  LIBRARIES $X AUSTRALIA+                      0002
 5.  LIBRARIES $X AUTOMATION*                     0004
 6.  LIBRARIES $X AUTOMATION $X CONGRESSES*       0002
 7.  LIBRARIES $X BIBL*                           0001
 8.  LIBRARIES $X CALIFORNIA*                     0002
 9.  LIBRARIES $X CALIFORNIA $X PERIOD*           0001
10.  LIBRARIES $X CALIFORNIA $X PERIOD+           0001
```

Of note here are the meanings of the special characters $X, *, and +. The $X is the MARC subject heading subfield $X and is used to identify a topic subdivision of a subject heading. In the Santa Cruz file (from which this example is taken), $X is a default value and will be used to identify geographic and chronological subject subdivisions as well as topical. The symbols * and + correspond to the MARC ₣ (end-of-field) and Ꞧ (end-of-record) signals.

Also of note are the double index file entries for lines 3-4 and 9-10. It is the current practice of CIMARON not to ignore the MARC ₣ and Ꞧ signals nor to treat these as logically equivalent codes. Consequently, the same subject heading will be entered twice if it occurs both as a final and a non-final field in two or more master file records. This practice does not effect search requests, and will probably be eliminated from future versions of CIMARON.

3.2.3 Advance the Display

In order to move the display, the user may specify a new Key Value (e.g. G 'ECONOMICS') or he may move forward in the file a fixed number of terms by entering:

```
FXƀ        (e.g. F 5 or F 10 or F 100)
```

where Xƀ is any number followed by a blank space. The display will then be advanced by X terms. When the next screen is displayed, it should be noted that the lines (i.e. index file entries) will be numbered consecutively from 1-10. To move backward in the list, only the G command may be used. WARNING:

Only numeric values of less than 100 should be used since the intervening terms must be retrieved and counted serially in order to maintain correct positioning. For larger moves use the G command.

3.2.4 Save an Index File Entry

Since the ultimate goal of browsing is to collect candidate terms for CIMARON2 search requests, BROWSER2 allows users to save index file entries for later use. This is done by transferring entries from the index file display to a special Save List or Save Area. The command to transfer index file entry to the Save Area is:

S Xb̸   (e.g. S 5b̸)

where Xb̸ is entry number (from 1-10) of t'e term to be saved. A number larger than 10 will result in a syntax error. The result of this command is to add the selected term to the end of the Save Area List.

3.2.5 Display the Save Area List

In order to review the current contents of the Save Area List, the user enters the command

D.

This causes the Save Area List to be displayed. The Save Area List looks exactly like an index file display. That is, each line is numbered consecutively from 1-10 and contains the term and its count. In order to return to the display of index file entries, the command <u>D</u> is re-entered.

3.2.6 Remove a Term from Save Area List

Frequently, the Save Area List may need to be pruned. In order to do this the following command is used:

R Xb̸   (e.g. R 5b̸)

where Xb̸ is any number from 1-10 followed by a blank. The command results in the deletion of the Xth term from the Save Area List.

3.2.7 Print

The Information Processing Laboratory as yet does not have a direct facility for producing printed versions of selected terminal displays. In order to accomplish this useful function, the line printer of the 360/40 at the Campus Computer Center is utilized. The command to create a hard copy version of a BROWSER2 display is:

H.

The command results in the printing of the current display, whether that display currently consists of index file entries or of the Save Area List.

3.2.8 Close or Exit

Two exit options are available. The current index file may be closed. The command for this is

```
//CLOSE.
```

This will terminate examination of the current index file and will re-initiate the question:

```
SPECIFY FILE NAME.
```

At this point another file may be opened for browsing. Closing a file does not affect the contents of the Save Area List.

To leave the program entirely and return to TMS the command

```
//EXIT
```

is used. This results in the termination of all BROWSER2 operations, including the purging of the Save Area List.

3.2.9 Display the Available Commands or File Names

If the user would like to review the commands available, he may enter:

```
//HELP.
```

This will result in a redisplay of the page defining the commands and their uses. In a similar manner, the available file names will be displayed in response to the user's command:

```
//LIST.
```

With these commands at his disposal, the user can master the use of BROWSER rapidly.

4. USERS' GUIDE TO FILE BUILDING

4.1 Overview

A core set of three ILR data base programs, FILOR, ZODIAC, and PAX, and an IBM utility sort are required to establish the three-level linked file structure which is searched by CIMARON2, the on-line retrieval routine. The three-level file structure that is established on disk consists of a search key file, an intermediate address file, and a master file. An index-sequential file is at the highest level and each of its records contains a search key and an address (link) to the next level file. If but one master record is referenced by a given search key, the associated address points to that master record. Otherwise, it points to a record in the intermediate address file. At the second level is an address file which has sets of addresses of master file records associated with a given search key in the level one file. At the third and last level of the file structure is the master file itself. It consists of a sequential array of master bibliographic records stored in a direct-access file. Any given record in this file can be accessed by the address obtained from the other files.

By dividing the file construction operations into separate, small, functionally oriented routines, the set of routines can be used flexibly in order both to carry out the file construction operations on a wide variety of files and to provide a wide variety of indexing to the individual records. This is not to say that any file could be utilized readily without modification of the routines, but rather that when modifications are required, it is a straightforward matter to identify the affected component routine and the nature of the change that would be required.

For example, ZODIAC, the routine which searches master bibliographic records and creates the individual search keys desired, presently is organized to obtain the master records from a random access disk file via an intermediate index file. If it were desired to obtain search keys from records stored in a sequential file for which no intermediate index were available, it would be necessary to modify only that segment of ZODIAC which is concerned with obtaining the next logical record from the master file.

As another example, if it were desired to utilize bibliographic records which are not in the MARC structure, it would not be necessary to modify any of the routines in the file building system except ZODIAC. It would be necessary to develop . new routine to replace ZODIAC since it is heavily dependent upon the MARC structure. And finally, only the display "CSECT" of CIMARON2 would have to be modified in order to store, search, and retrieve these records appropriately.

4.2 Creation of the Bibliographic Master File

The first step in the file creation sequence is performed by the program known as FILOR. This program takes as input a file containing source master records and provides two files as output: the first is a direct-access master file which ultimately will be the third level of the file structure, and the second is a sequential finder file in which each record contains the master file record accession number and the disk address at which that record can be found. At present the input file is defined as a sequential file of variable length master records, either blocked or unblocked, with standard IBM convention on placement of the record length information. However, with minor modification, the program could run on fixed length records as well.

The first output file is a direct access file and the variable run parameters in this file are the block size (BLKSIZE) and the logical record (LRECL) length.* These two parameters in association with the UNIT and VOLUME parameters would enable this file to be established on a variety of direct-access secondary storage media.** At present, if a master record will not fit within a physical block, it is segmented and stored in two contiguous blocks. The finder file, the other output file produced by this program, is a sequential file of blocked records. The block size is again a variable parameter, as are UNIT and VOLUME. This file thus can be established on any sequentially accessible storage medium, and in general a tape file is used.

An important characteristic of the direct-access master file created by this program has to be mentioned here. A logical record in this file can be laid out across a block boundary.*** This has been done with the view of optimizing the packing density in this file. Thus, routines which attempt to retrieve records from this file would have to make use of "splicing" procedures**** for split records. Split records are those having a first part at the end of a given block and the second part at the start of the next sequential block in the file. Information about the size of each part in the two different blocks can be obtained easily from the three-part disk address which has been described earlier. The combination of track number, track offset, and record length, in conjunction with a knowledge of the capacity of each track or block is sufficient to determine the sizes of the two parts of a split record.

Each record of the finder file consists of eighteen bytes (see Figure 13). The first six bytes contain the record (accession) number in EBCDIC, and the next twelve bytes constitute what is known as the pointer field. This field essentially consists of an eight-byte disk address, in addition to type and flag information. The type and flag information in each finder file record an attempt to standardize the file types and the nature of the content of each file into two or three classes so that future programs can obtain dynamically the type of the file and the nature of its content and invoke specialized retrieval routines. In detail then, the first six bytes give the accession number; the seventh byte in the finder file record is a code indicating the type of file (e.g., index sequential, direct-access, etc.); the eighth byte in the record is a code indicating the type of content. Currently three types of file content are defined: key-type content, address-type content and data-type content. The master file for example, would be a file with data-

*For reasons of storage efficiency, it is recommended that BLKSIZE equal LRECL equal track capacity on disks and drums, i.e., the record format is fixed (RECFM=F).

**Including tape devices with a block-skip feature.

***In the terminology of IBM access methods such records would be known as spanned records.

****May now be available in BDAM, RECFM=FS.

type content. The ninth and tenth bytes in the record constitute a two-byte file code. Currently the letters MF have been chosen to indicate the master file. Other codes are for attributes, for example AU for authors, TI for titles, SU for subjects, etc. (These codes are established by PAX.) The remaining eight bytes constitute the true disk address of the master file record. Of these, the first four bytes give the relative track number in binary, the next two bytes give the offset into this track at which the master record begins, and the last two bytes give the length or extent of the master record. The four-byte track number has been designed towards making use of the IBM direct block addressing facility. This facility requires a three-byte relative block number, which in our case happens to be the relative disk track number.* The fourth byte could be used in the future to indicate the relative unit on which the file is to be found. This would handle the case of a very large file which spans a number of disk units. However, CIMARON currently is established to accept only the value 0.

FIG. 13: FINDER FILE RECORD FORMAT

File Name (controlled by JCL of FILOR)
Type of File: Sequential
Record Length: 18 bytes

Data Type	Field Name and Position
EBCDIC	Accession number (1-6) Pointer field (7-18)
EBCDIC	Type [of the file referred to] (7) Values: 1, index sequential $\boxed{2}$,** direct access 3, sequential
EBCDIC	Type of content (8) Values: 0, key 1, address $\boxed{2}$, data
EBCDIC	File code (9-10) Values: $\boxed{\text{MF}}$, master file AU, author index etc.
Binary	Track location (11-13)
Binary	Unit number*** (14) Values: $\boxed{1}$, 1st 2314
Binary	Offset (15-16) location of beginning of record
Binary	Record length (17-18)

*If the master file were established on tape, this would be the relative block number.

**□ indicates the standard values set by FILOR.

***Unit number is provided in order to allow storage and access to multiple disk units. To be put into effect, however, additional unit designations must be included in CIMARON Table.

FIG. 14: ZODIAC RUN SETUP

Job Control Language (JCL):

```
//B5844JAZ  JOB  (5844,15,50,00),'ILR-CUNNINGHAM',MSGLEVEL=1,CLASS=L
//GO      EXEC  PGM=ZODIAC2
//STEPLIB DD   UNIT=2314,DSN=ILR.BATCHLIB,DISP=SHR
//GO.SYSUDUMP  DD  SYSOUT=A
//GO.LONEACC DD UNIT=2314,VOL=SER=ILR02,DSN=ILR.SCAC3,DISP=OLD,
// DCB=(RECFM=FB,BLKSIZE=1800,LRECL=18)
//GO.MASTERR DD UNIT=2314,VOL=SER=(ILR03,ILR05),DSN=ILR.SCMF2,
// DCB=(DSORG=DA,RECFM=F,BLKSIZE=7294),DISP=OLD
//GO.PRINT  DD  SYSOUT=A,DCB=(RFCFM=FB,LRECL=25,BLKSIZE=750)
//GO.AUTHORS DD UNIT=TAPE,DCB=(RECFM=FB,LRECL=94,BLKSIZE=1692,
//  TRTCH=C,DEN=2),LABEL=(1,BLP),
//  VOL=SER=(4117R,4118R),DISP=(NEW,KEEP),DSN=SCAUTH
//GO.TITLES  DD  UNIT=TAPE,DCB=(RECFM=FB,LRECL=94,BLKSIZE=1692,
//  TRTCH=C,DEN=2),LABEL=(1,BLP),
//  VOL=SER=(4119R),DISP=(NEW,KEEP),DSN=SCTITLE
//GO.SUBJECT DD UNIT=TAPE, DCB=(RECFM=FB,LRECL=94,BLKSIZE=1692,
//  TRTCH,DEN=2),LABEL=(1,BLP),
//  VOL=SER=(4121R,4122R),DISP=(NEW,KEEP),DSN=SCSUBJ
//GO.CARDIN    DD     *,DCB=BLKSIZE=80

AUTHORS  100.4,110.4,111.4,700.4,710.4,711.4

TITLES   130.4,240.4,245.4,440.4,730.4,740.4,840.4

SUBJECT  600.4,610.4,611.4,630.4,650.4,651.4,660.4

/*

//
```

INPUT FILES:

Master file - DSNAME=ILR.SCMF2 refer DD=MASTERR

Finder file - DSNAME=ILR.SCAC3 refer DD=LONEACC

Controls - none at present

OUTPUT FILES:

Authors - DSNAME=SCAUTH refer DD=AUTHORS

Titles - DSNAME=SCTITLE refer DD=TITLES

Subjects - DSNAME=SCSUBJ refer DD=SUBJECT

Controls - Through card input refer DD=CARDIN

The control input for the program FILOR is indicated on cards as shown in Figure 15. The information supplied on these cards is as follows: the number of records that are to be appended to the master file, the relative track number, and the track offset at which the records are to be appended. In the case of a master file which is being established on disk for the first time, the track number and offset would be zero, indicating it is being appended "from the beginning." In the case of update runs, track number and offset in the master file would be obtained from the run statistics of the previous run, and this would be input as control to the program so that the file is appended at the right point.

4.3 Extraction of the Index Information

The next program in the file creation sequence is known as ZODIAC. This program has the function of extracting various fields from MARC records as attributes of the record, on which index files will be subsequen .y established. The program is definitely tied down to the MARC II format* in that each record is expected to have a MARC structured leader and directory through which the variable fields are obtained. These variable fields which will be used to establish the index files are selectively extracted by the program. This program has two input files and a variable number of output files besides the controlling input which is supplied on cards. The two input files are the same files which are produced by FILOR in the first step of the file creation process viz. the sequential finder file and the direct-access master file. The controlling tables which are set up as a result of reading in the parameter cards determine the number of sequential output files created and their content (see Figure 14).

The structure of any record in an output file is the same. It's a 94-byte record of which the first eighty bytes contain the variable field which has been extracted for the purposes of establishing an index. Since only eighty bytes are allowed** there might be cases where the field was truncated. The remaining fourteen bytes in each record consist of two parts. The first two bytes represent the MARC tag (in binary) identifying the type of variable field, and the last twelve bytes are the pointer field (see Figure 13) picked up from the last twelve bytes of the finder file entry for this particular MARC record. The construction of an output file record in this manner makes sure that any variable field which is extracted from a given MARC record is firmly associated with an address 'pointer' to the MARC record itself and the tag identifying this variable field in the MARC record.

4.3.1 Parameter Control

A parameter card consists of the name of the output file followed by a sequence of three-digit tags, delimited by commas. The tags indicate the variable fields that have to be extracted from each MARC record and routed to a given file whose name is supplied in the card. For example (see Figure 14), a parameter card which could be supplied as controlling input to ZODIAC may

*See "Specifications for Magnetic Tapes Containi Monographic Catalog Records in the MARC II Format," in Books: A MARC Format, Washington, D.C.: Library of Congress, Information Systems Office, April 1970.

**This is a reprogrammable parameter.

FIG. 15: FILOR RUN SETUP

Job Control Language (JCL):

```
//B5844JAF  JOB  (5844,15,50,00),'ILR-CUNNINGHAM',MSGLEVEL=1,CLASS=L
//GO   EXEC  PGM=FILOR2,COND=COND=EVEN
//STEPLIB  DD  UNIT=2314,DSNAME=ILR.BATCHLIB,DISP=SHR
//GO.SYSUDUMP  DD  SYSOUT=A
//GO.CARDIMIN  DD  UNIT=TAPE,VOL=SER=(3717,3760,3780),LABEL=(1,BLP),    X
//               DCB=(RECFM=VB,LRECL=2048,BLKSIZE=3600,TRTCH=C,DEN=2),  X
//               DISP=(OLD,KEEP)
//        DD UNIT=TAPE,VOL=SER=(3725,3766,3787),LABEL=(1,BLP),          X
//               DCB=(RECFM=VB,LRECL=2048,BLKSIZE=3600,TRTCH=C,DEN=2),  X
//               DISP=(OLD,KEEP)
//GO.LONEACC  DD  DCB=(BLKSIZE=1800,LRECL=18,RECFM=FB),UNIT=2314,       X
//  VOLUME=SER=ILR02,DISP=(NEW,KEEP),DSN=ILR,SCACC2,                    X
//  SPACE=(CYL,(15,1),RLSE)
//GO.MASTERR  DD  UNIT=2314,DISP=(NEW,KEEP),                            X
//               VOL=(,,,2,SER=(ILR03,ILR05)),                          X
//               SPACE=(CYL,(199,15),RLSE,CONTIG),DSN=ILR.SCMARC2,      X
//  DCB=DSORG=DA

/*

//
```

INPUT FILES:

Master file - DSNAME=   refer DD=MASTERR
Controls   - none at present

OUTPUT FILES:

Master file - DSNAME=ILR.SCMARC2 refer DD=MASTERR
Finder file - DSNAME=ILR.SCACC2 refer DD=LONEACC

consist of the word AUTHORS in the first seven columns of the card followed by the tag 100 beginning at column ten.* This would indicate that all variable fields associated with tag 100 (which identifies a personal author entry) be routed to an 'authors' file. The digit following the tag specifies an initial offset in the variable field. This is tc skip over binary indicators and codes which may occur at the start of the field. Each record in the output file SCAUTH would consist of a personal author name (blank filled) in the first eighty bytes, followed by the tag 100 in binary in the next two bytes followed by the twelve-byte pointer field (see Figure 16).

FIG. 16: ZODIAC Internal Control Tables

NAME	maximum of twelve eight-byte entries, one entry for each output file
CALTEX	maximum of twelve one-byte entries
MARCEL	850 one-byte entries, one for each possible MARC tag; each byte is a possible index to an entry in the NAME table
TAGOFF	850 one-byte entries; each byte gives an offset from the beginning of the field, to skip fixed length control (e.g., $a)

4.3.2 ZODIAC Control Tables

It might be instructive to describe the tables which are used to drive this program. There are four important tables in this program: NAME, CALTEX, MARCEL and TAGOFF. These tables are summarized in Figure 16. The first is known as NAME, and this table can have twelve eight-byte entries. This is a table of the twelve possible output file names each of which can be eight characters long. The next table is known as CALTEX, and this can hold twelve eight-bit flag entries. The next table is known as MARCEL, which has 850 positions - each of which is one byte wide. This table reserves one entry for each possible MARC tag. MARC tags currently run from 000 to 850; thus we have 851 positions in this table. The table is initialized to blanks and after the reading in of the parameter cards, a given entry in this table contains a one-byte index into the NAME table. This defines a given tag as being required in a given output file. A table which is parallel to this table is known as TAGOFF, which also consists of 850 one-byte entries. Each entry in this table gives an offset from the beginning of the variable field in order to skip over fixed-length control fields at the head of the variable field. The offset table is set up in parallel with MARCEL, as a result of reading in the parameter cards. On the parameter card each tag is followed by the offset, which is supplied within two digits and delimited from the tag by a period. This two digit offset indicates the number of bytes that have to be skipped over from the beginning of the variable field in order to get at the data which is required in the output file. For example, we currently skip over the sub-field delimiter and the sub-field code at the head of variable fields. The other sub-field delimiters and the sub-field codes within the variable field itself will be carried as part of the output file. The table known as CALTEX is associated with the programming logic.

*The file name can be up to eight characters long in accordance with O.S. conventions. Column 9 is ignored. Tags must begin in column 10.

This table is initialized to zero and indicates by a non-zero entry the fact that records have been output to a given file previously. Thus, on the first time that a given tag is encountered in a MARC record and found to be required in a given output file, a special routine finds zero entry for that file in CALTEX and obtains working storage for that file, opens the file and performs other initialization procedures. CALTEX is a table of flags which indicates whether the initialization procedures have been performed or not.

4.3.3 Processing Sequence

We might end by briefly describing the typical sequence of operations that take place on reading in a particular MARC record. First a record is read in from the finder file and the portion of the finder file record which gives the record address is used to issue a read to the direct-access master file. This results in bringing in a particular MARC record into main storage. Next, the MARC record directory is sequentially scanned from beginning to end and the tables simultaneously consulted. As each tag is encountered the table MARCEL is looked into and a non-blank entry indicates that the variable field associated with this tag is required in a given output file. The name of this output file is indirectly known via the index quantity in the current entry of MARCEL. By making use of this index the flag table, namely CALTEX, is indexed and it is determined whether this file has been initialized for output or not. In the event that it has not been initialized for output, initialization procedures are performed and then one goes on to the subsequent portions of the routine. These subsequent portions pick up the relative address of the variable field from the current entry in the record directory and move the variable field out to the output record buffer where a 94-byte logical record described earlier is constructed and written out to the associated output file. This procedure is repeated by going back to the finder file and finding the address of the next MARC record, after all the directory entries in the current MARC record have been scanned.

4.3.4 Program Constraints

There are two limitations in the number and nature of output files that can be created in one run. First, the number of output files that ZODIAC can create is variable up to a maximum of twelve. This is a limitation of the table-structure in the program because the first step in the program is to read the parameter cards and create tables which are subsequently used to drive the program in its variable field extraction procedures. The second limitation is that there must be an exclusive partition of the tags across the various output files, in other words a given tag cannot appear in more than one output file-defining parameter card. This means that a MARC variable field can be routed to one and only one output file.

The execution of this program is also controlled by the finder file in the following manner: as many records are brought into main storage and analyzed as there are entries for them in the finder file and the finder file is accessed sequentially from beginning to end. A possible future control on this program would be to define contiguous subsets of the master file to be analyzed by ZODIAC, through the incorporation of record skip and record count controls.

Currently the program is tied to the format of the finder file (18-byte records) and also the format of the output files (94-byte records). By reassembly, the length of the output records can be changed, in order to change the length of the key field carried in the output records. This length can be anywhere from 1 to 256 bytes, since the variable field is eventually to be used as the key in an index-sequential file, and the O.S. limitation on the size of this key is 256-bytes.

4.4 Sequencing the Index Data

The next step in the file creation procedure is executed by the IBM sort utility. Each of the output files produced as a result of a run of ZODIAC are passed through the sort utility, which sorts the records so that the key portion (the first 80 bytes) are in alphabetical sort order. Each run of this program takes one input file, ›rts it and produces one output file, which is then ready for the final step in the procedure.

The controls supplied to the IBM sort utility are as follows: the record length, the offset from the beginning of the record to the sort key in each record, the length of the sort key, and the sort order, namely ascencing or descending. The sort utility also has provision to specify the sort key in two parts. This in fact is being done currently with the records output from ZODIAC. As mentioned earlier, each of these records consist of 94-bytes, the first 80 of which have the variable field and the last twelve contain the pointer field, which contains the disk address of the master file record. The sort is currently being performed primarily on the variable field and secondarily on the twelve-byte pointer field. This insures that any collection of pointers in the level two address file of the file structure associated with some key in the level one file, will be in ascending order of master file address. This buys a little efficiency in the running of the retrieval routines. One of the first things that is done on retrieving a list of addresses from disk is to put them in a sort order, so that comparisons can be done between two lists. So by specifying this secondary sort key one can save the initial sort on reading in a list of addresses from disk.

4.5 Creation of the Index Files

The final step of the file creation procedure is performed by a program called PAX. This program has one input file and produces two output files. The input file is a sequential file and it is the output of the IBM sort utility. The two output files produced by this program are respectively, the level one index-sequential access file and the level two direct-access address file. If a pair of these files is established for a given attribute in a record colle^t^ion, it will enable the master file collection to be searched via this attribute.

The process performed by the program PAX and indeed much of its logical structure is quite similar to that of FILOR. The points of difference lie in the format and content of files handled by this program. The level one index-sequential file consists of records which have two important parts. The first part is the key and this key will be the argument for searches performed on this file. The second part of the record consists of a twelve-byte pointer field, which will point to a collection of twelve-byte pointer fields in the level two address file. The length of the record keys of this first level file is variable up to a maximum of 80. On loading different files, this can be varied without reassembly of the program, since the key length is supplied as a PAX run-time parameter. After the file is opened (e.g., by CIMARON) this parameter is available in the data control block for the file and thus files can be created with records different length and the using program can be adjusted to the key length of the specific file.

Currently the two-byte binary tag which is carried in the input records does not appear in the index-sequential file. At a later time, it may be used to further 'refine' attributes (to the tag level) during search.

One of the important functions performed by PAX is to make sure that only one index-sequential record is created for each unique key in the input file. This is established on the basis of a comparison between the 80-byte variable field portion of the current input record and the corresponding field in the previous record. At the point when a mismatch is detected the program will create a new record in the access file - that is the level one file. This record would consist of a key constructed from the variable field portion of the previous record followed by a twelve-byte pointer field which points to the location in the level two file at which can be found a sequence of twelve-byte pointer fields identifying all the MARC records in which this particular variable field appeared.

It is important to note that the structure of the pointer field is uniform throughout the system, and it is by this field that links are established across the levels of the file structure. Specifically, the pointer field in a record in the level one file establishes a link to a series of pointer fields in the level two file. Each pointer field in the level two file establishes a direct link to a master record in the level three file. This briefly, is the file structure employed to search master file records via indexed attributes.

The creation of entries in the level two file proceeds in parallel with the reading in of input records. As each input record is read in and its sort key found to be the same as that of the previous record, an additional entry (a twelve-byte pointer field entry) is made in the level two direct-access file. This additional entry is in fact the pointer field in the current input record. At the end of a sequence of identical sort keys on input records, it is time to create a new record in the level one file for the collection of input records which have the same variable field. The address at which a sequence of master record addresses can be found associated with this collection of similar variable fields is entered in the level one file.

Where there is but one master record linked to a given sort key, the level two file is bypassed. Instead, PAX establishes a direct link from the record in the level one file to the master record in the level three file, in all cases where the sort key in the level one file is uniquely associated with a single master file record in which it appears. This enables one to bypass both the construction and subsequent reading of a record in the level two file during search. Thus, the speed of the search process is increased for such keys.

In the future, it may be possible to dispense with the level two file altogether. In this improved file structure, all of master file addresses associated with a key would be found in the level one file entry for that key. However, this will be possible only when the IBM operating system supports variable-length records in index-sequential files. Currently only fixed-length records are supported in these files.