

AN INTERACTIVE BIBLIOGRAPHIC REFERENCE SYSTEM

by

KATHLEEN ANN MILLER

BA, University of Southern Colorado, 1973

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE


Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

-1984-

Approved by:


Major Professor

LD
2668
R4
1984
M54
C. 2

ACKNOWLEDGEMENTS

My heartfelt appreciation is extended to Dr. Elizabeth Unger. She is a fine person, a conscientious educator, and a dedicated computer scientist. It has been my privilege to have known her and worked with her through this project and in several classes.

I want to recognize my committee members, Dr. Rod Bates and Dr. Rich McBride, for their assistance. Acknowledgement, too, goes to Harvard Townsend for his help and advice on OS/32 and UNIX.

I want to extend a loving thanks to Frank Zacharias. He has continually been a source of optimism, perspective, and support for me. Finally, thanks to Brindie for her sense of humor and for getting me into computers.

CONTENTS

	Page
Acknowledgements.	ii
List of Figures	vi
Chapter 1	
Introduction.	1
Comparison of System Models	
Information Retrieval Systems	4
Data Base Management Systems.	4
Management Information Systems.	6
Question-Answering Systems.	6
Decision Support Systems.	7
System Functions to Consider	
Considerations Related to the User.	8
Considerations Related to Hardware/Software . .	10
This Problem	
Desired System Capabilities	12
Which System Model?	15
System Software Alternatives.	16
Chapter 2	
KGS Development	18
Operations	
Create.	19
Update.	19

	Page
Delete.	19
Retrieve.	20
Information	21
Summary of KGS Capabilities	22
 Chapter 3	
Data Base File Structure.	23
Modes of Operation	
User Mode	
Edit (KGSEDT).	29
Extract (KGSEXTRT).	31
Information (KGSINFO)	35
File Maintenance Mode	36
 Chapter 4	
Introduction.	40
Will It Work on Our Computer?	40
Can You Really Trust Documentation?	41
How Do You Convert Fortran VII to Fortran 77?	43
How Do You Convert OS/32 MT to UNIX?	44
What Test Procedures Should Be Used?	44
 Chapter 5	
Introduction.	46
Fortran VII to Fortran 77 Conversion	
Character Type Declaration.	46
Six-Character Labels.	47

	Page
Decode Routine.	47
Unit Reassignment	47
File Specification.	48
Inquire(SIZE=	48
Formatted Files	49
SHARE Control	49
File Status "RENEW"	50
Date and Carcon Routines.	51
END and ERR	51
OS/32 MT to UNIX Conversion	
Cases	54
Compile and Link.	54
Load.	54
Enhancements.	55
KGS - The Final Product	56
Chapter 6	59
Appendix	
KGS User's Guide.	63
KGS Source Code Listing	74
Bibliography.	155
Abstract	

LIST OF FIGURES

	Page
1. Inter-relationships of System Models.	5
2. KGS Data Base File Structure.	24
3. KGS Master File Record Format	25
4. Dictionary Record Formats	26
5. Index Record Formats.	27
6. KGS User Mode - Program Calling Structure	30
7. Sample KGS Output	32
8. KGSINFO Sample Output for Authors	37
9. KGSINFO Sample Output for Keywords.	38
10. KGS File Maintenance Mode-Program Calling Structure	39

Chapter 1

Introduction

It is an extraordinary era in which we live. It is altogether new. The world has seen nothing like it before. I will not pretend, no one can pretend, to discern the end; but everybody knows that the age is remarkable for scientific research into the heavens, the earth, what is beneath the earth; and perhaps more remarkable still is the application of this scientific research to the pursuit of life. The ancients saw nothing like it. The moderns have seen nothing like it until the present generation. . . . The progress of the age has almost outstripped human belief. Daniel Webster, 1847

As we are stepping into a new age of technology, we are plagued by an old, ever-expanding problem --- information explosion. Written materials are published faster than any single human could even categorize. The number of articles published in periodicals, alone, nearly doubled from 1,985,000 in 1960 to 3,780,000 in 1970 (HEA78). Previous estimates of rate increases were only for doubling every fifty years (SAL83). Today, more than ever, our lives, in both a global and private sense, involve the collection of information. How can we deal with this voluminous labyrinth of text?

Established paper-based information retrieval techniques are slow and ineffectual in tackling our present-day information problem. Nevertheless, paper-based systems are still the standard for most professionals (scientists, researchers, educators, writers, etc.). It is clear that

many people do not even rely on paper trails as clues to the whereabouts of books and articles. If you ask for information on a given subject from one of these professionals, you are apt to become engaged in an adventure through piles of dust-covered pages or shelf upon shelf of orderless books. "I'm sure it had an orange cover and the author started with 'Pyl' and I don't think it was the one that I had under that plant..." Even inquiring about your subject with a professional who has maintained meticulous card files of entries (including title, author, several subject descriptions, etc.), alphabetically ordered by author, may or may not be very effective, depending upon the ability of that person to associate subject with author. If there has been a break-down in that associative memory, then you may be treated to thumbing through file boxes, reading the subject lists on each card.

Even our nation's libraries remain paper-based in retrieval techniques, though they have made it easier for us (than our card-file person) with indexes by author, title and subject. This means at least two complete duplications of the information on cards to accomplish this, however. Again, we are thumbing through endless ranks of cards. Researching a specific topic in the periodical room is even more difficult since what you find is an index to an index to an index, at which point you may still not know if the paper is even in the library.

What libraries and professionals need is an up-to-

date listing of all materials available on-site, which may be indexed in any number of manners . . . author, title, subject (or combination of subjects), related subjects(s) . . . and quickly! (Time is, indeed, money.) Again, how can we deal with this?

Luckily, some sophisticated solutions are available through advances in computer technology. Over the past 15 to 20 years the trend has been a doubling of computational capabilities every 3 to 4 years, together with a decreasing cost of those capabilities (SAL83). Efficient access techniques and cheap, improving storage facilities combine to provide an excellent environment for the electronic manipulation of text-related materials. Reference materials may be managed interactively, allowing queries by author, title, or subject key(s). In a few minutes, our professional could have an entire listing of all related, on-site materials and their location. A system like this could be beneficial to faculty members conducting joint research projects, individual writers of papers and grants, as well as classes of students.

Many models for information organization and retrieval have been developing in recent years, including systems such as Information Retrieval (IRS), Data Base Management (DBMS), Management Information (MIS), Decision Support (DSS), and Question-Answering (QA).

How does our professional choose a system to suit particular reference needs? In order to approach this

question, we first take a general look at the major system models and then at the functions which we want to consider for any system we choose.

Comparison of System Models

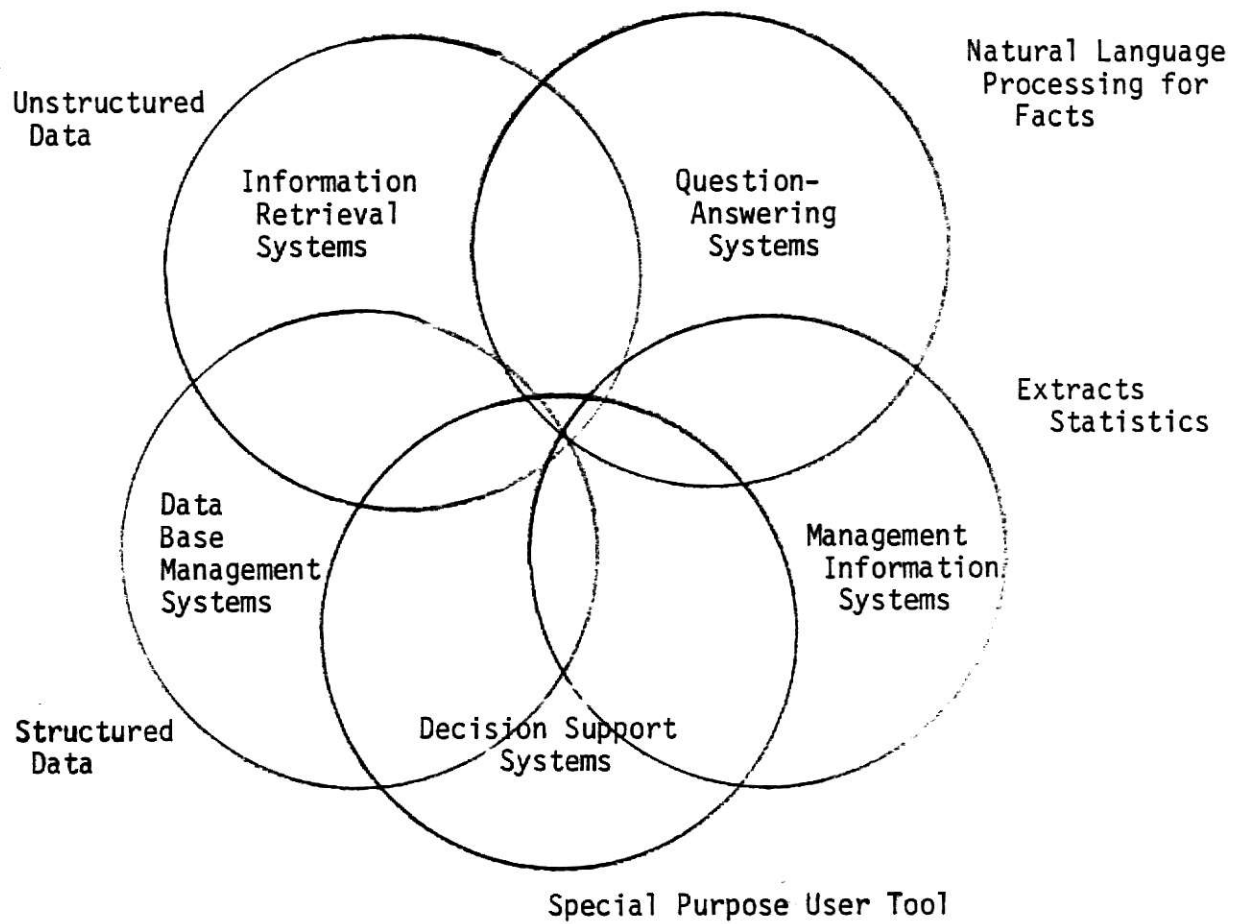
Information Retrieval Systems

The basic element of the IRS is information in unstructured natural-language (English) form, such as complete documents and references. Though some indexes may be stored in structured (i.e., static field length) records, the actual information is stored in an unstructured format, due primarily to the variable-length nature of documents. This kind of system usually requires excessive serial searching through large numbers of records, so may tend to require long access times. IRS in the paper world might be related to the activity of a person leafing through a book to find required information (with the assistance of a table of contents and index).

An IPS is capable of satisfying approximate queries, such as giving all information related to a specific English word root (e.g., comput***). The result of the query is a selection of information elements.

Data Base Management Systems

Though similar to an IRS in the general concept of retrieval, a DBMS is typically composed of structured data elements, such as numerical entries and character strings, both of fixed length. These data elements are stored in



Inter-relationships of System Models
-Figure 1-

tabular form and may only be accessed by an exact-match query, unless special front-end software is provided that allows a user to utilize a synonym or word-root table or procedure that searches for all strings similar to the target string or which include the target string. The result of this query is a selection of data elements.

Management Information Systems

Many businesses depended on non-automated MIS long before automated DBMS allowed for quick access to large amounts of data. By analyzing a given data base, an MIS may provide useful information regarding the statistics of the data base. A good example of this is the statistics taken from the U.S. National Census, which allows us to discover totals, average and range of values on the U.S. population data base. These statistics are quite interesting, but more importantly, over time provide a sense of variation in values that allow us to make meaningful projections based on previous trends.

Question-Answering Systems

If you could address a computer through a keyboard with a question such as "How many days of rain did Phoenix receive in 1970?" and receive a response in natural language form, you would most likely be working with a Question-Answer System facility. The basic element of this system is the fact. These facts may be drawn from an IRS or DBMS or other systems. The important part of this system is the

intelligence necessary to interpret a user's question, decide upon the proper procedure for obtaining the desired fact, maneuver through systems to obtain the fact, and finally deliver that fact to the user in understandable form (English). Artificial intelligence technology holds great promise in this area and we can expect to see at least small, special purpose systems in the near future. At this point, however, QAS are mostly in the research and development stage.

Decision Support Systems

It is difficult to position DSS in a specific position in relation to IRS, DBMS, MIS and QAS since its placement is highly dependent upon a user's particular application requirements. Different users, understandably, have different areas of interest and different needs in terms of information required for making high-level decisions. DSS is, then, a special purpose system designed to meet one user's specific needs in supporting management level decision-making efforts. Though there may be some similarities between DSS and MIS, it should be noted that DSS may take statistics and derive tables, graphics, or other formats for the presentation of facts or information. Unlike the QAS, these fact-presentations are not answers to spontaneous questions, but are designated as system requirements at the time the software for the DSS is designed. Therefore, the DSS is more static in nature than the QAS. It is also more extensive than MIS because it may

draw facts from statistics. Special tools such as computer graphics systems may be incorporated in order to provide clear pictures of facts.

System Functions to Consider

The first step in the design of any computer system should be the analysis of the user population's requirements. Consider how this relates to our reference problem. We can define at least two fundamental requirements of our system:

- 1) The system must provide the user with the ability to enter and update stored references.
- 2) The system must provide the user with the ability to retrieve stored references.

Beyond these basic requirements, we can include additional requirements of the design that take into consideration special user and system (hardware and software) considerations that define this problem for a specific user.

Considerations Related to the User

In a professional setting, a more user-friendly system can realize a savings of time and money by improving performance. Therefore, special user requirements must be important considerations. In writing on office systems development, Dimitris Chorafas asserts that user performance relates to ease-of-use, transparency, response time and cost (CHO82).

Ease of use includes language, training, communications and effort expectations. Must the system

interact with the user in English or will simple character prompts suffice? What is a reasonable training period for a user to be fluent in the system's basic features? Is communication with the system accomplished with cards, keyboards or other entry devices (e.g., voice recognition, light pen, human touch) in a batch or interactive environment, or both? How much effort is involved in completing the tasks of information entry or retrieval? Are there alternative procedures for retrieval of a reference (i.e., author, title, subject(s), other)?

Transparency (data independence) is associated with ease of use issues. How much of the total system design (i.e., data format, storage organization, and distribution) does the user need to know and understand in order to use it effectively?

What is the minimum tolerable response time for this system? There may be several response time limits if there are a variety of communication environments.

Finally, what are the maximum time and money constraints in the development of this system? Will there also be any costs linked with usage? It is important to realize that cost constraints may be the determinant of final performance requirements specifications. There is always a higher cost associated with the more refined, user-friendly facilities. Therefore, all aspects of user-friendliness must be weighed against each other in light of cost constraints.

There is one more factor of performance that I will add to Chorafas' list: correctness. No matter how friendly a system is to the user, it is worthless if it does not accomplish the initial goal of providing accurate response to a user's request for information.

Considerations Related to Hardware/Software

User requirements must be met through both hardware and software support (CH082). The greatest balancing in our design-under-cost comes, then, between the user and the system's hardware and software capabilities.

In order to satisfy the ease-of-use requirements of the user a great range of sophistication of hardware/software may be suggested. The more user-friendly system may require extensive menu display, graphics or even voice synthesis capabilities. A less friendly system might include only the minimum of hardware/software to display simple one-character prompts, relying on the user to know what information is expected. Therefore, the degree of system sophistication relates directly to the amount of training the user must complete to be proficient in its use. A variety of communication environments will require more from a system - especially if it is to accommodate interactive devices among many users. System detail is also a function of user effort as well as transparency. Must the user know details of data format or does the system software enforce correct format upon entry?

If the user's minimum response-time requirement is

low, then hardware and software must be designed to meet that requirement. This may mean using a faster, low-level language, such as assembler, in order to provide fast program performance. It may also impose the requirements of using a special data organization. For slower response times, coupled with small amounts of data, linear unsorted file structures might be satisfactory. However, for faster response times and larger amounts of data, structures from ordered sequential or inverted files to faster pointer-based or other devices would be needed.

When we begin to discuss a system's provisions for correctness requirements, we must discuss issues of integrity, security and recovery. These issues also relate to basic data organization within a system, but may, in addition, necessitate physical access and device requirements. Integrity addresses the prevention of nonmalicious errors (ULL80). In order to insure correctness, we must be able to monitor the entering, updating and deleting of data, with the capability of preventing or correcting thoughtless mistakes (e.g., masked monitored screens with prompts). Security addresses the issue of access control by restricting users to access and/or modify only specified subsets of the data (ULL80). Security restrictions may be accomplished through software (e.g., passwords, read/write constraints), hardware (e.g., only special keys on special machines) or physical location access control (e.g., only special machines with physical

security constraints). Recovery involves the prevention of loss of information due to hardware malfunctions. Recovery can be accomplished through a wide variety of backup schemes, including data duplication, regular dumping onto magnetic tape and other methods.

This Problem

The faculty of the Computer Science Department at Kansas State University is actively involved in both educating students and researching current topics in computer science and related fields. Accordingly, there is a need for a bibliographic reference system in order that faculty might:

- 1) direct students to available materials related to class or project topics,
- 2) maintain an organized base of research materials related to an individual faculty member's work and
- 3) provide faculty with access to other faculty members' bases of research materials in order to facilitate joint class or research project work.

These three points constitute our overall goal for a system. In the next section, the desired system capabilities are defined in more detail, followed by a section which examines the criteria for choosing a system model that might best fulfill those capabilities.

Desired System Capabilities

If users find the system difficult to use, then they

will be less apt to use this tool in their research efforts. To be a useful tool, therefore, system interaction must be easy for the user. The language must be simple and primarily in English. Simple logical commands should form the basis for the user-to-system language needed to enter, retrieve, update and delete reference items. Training time may be greatly reduced by having system prompts incorporated into crucial branching points within the system programs (e.g., "Do you wish to ENTER, RETRIEVE, UPDATE or DELETE reference items?"). In addition, a HELP facility would be useful for providing important information to first-time users. Communications between user and system might, ideally, be voice-based, but more realistically could adequately be accomplished with a terminal keyboard and screen. Since new users require more help than experienced users, the degree of effort in use of the system is dependent upon the skill level of the user. Therefore, the system should include the ability to distinguish between the beginner and the pro and adjust the mode of interaction, accordingly.

The system should provide the user with a high degree of data independence. The user should not need to be knowledgeable of data organization or formats. Even though the users in this case are highly competent Computer Science professionals, they should be able to operate the system without a sheet of acceptable data formats.

System response time should be short (probably less

than ten seconds). Therefore, an interactive multi-user system is needed. This allows users to communicate from their own terminals at any time, regardless of the number of other system users, and expect a minimum of waiting for a response.

Monetary cost constraints are, necessarily, a limiting factor at this time of tight budgeting within the education community. Thus existing hardware (a Perkin-Elmer 8/32 minicomputer, operating with a Unix operating system) must be utilized. The Perkin-Elmer hosts a variety of peripheral devices, including a terminal in each faculty member's office and several printers within the department. The limitation of funds also means little money for the purchase or development of software. The time constraints on this project are "do it as soon as you can."

Finally, the important factor of correctness is considered. Security measures should restrict access to faculty members. After the initial entry of many reference items, it is estimated that information entry will be limited to a small number per day, which would make daily dumping of all references onto magnetic tape a sufficient back-up/recovery technique. In order to preserve the integrity of the information, the system hardware/software must provide a means of accurately retrieving and updating reference items with checks for correct data values and formats. Also, because the reference system will be accessed by multiple-users, there must be facilities to

prevent or eliminate update anomalies.

Which System Model?

In examination of the desired capabilities of our reference system, it becomes evident that this is a special-purpose system that deals with English-form data of fairly fixed length (i.e., author, title, descriptors). We want as high a degree of user-friendliness as possible at as low a cost as possible, on existing hardware.

Upon re-examination of the system models available, we eliminate Question-Answering Systems from our list of feasible systems since it is still in the research and development stage and would likely require more time than our "as soon as you can" constraint will allow.

Information Retrieval Systems have the drawback of long response times. Also, IRS are more suited to extensive widely-varying lengths of textual materials, which are not the type of materials (reference items) with which we want to work. (This kind of system would be more suitable if we were going to store entire articles or chapters of text.)

Management Information Systems function more as interpreting or analytical agents than as querying agents on a base of data or information. Hence, this system, too, seems inappropriate for our needs.

Finally, we are left with Data Base Management Systems and Decision Support Systems. DBMS provide the storage facilities that allow quick access of our reference items through queries. The DSS reflect the special-purpose

nature of our desired reference system. Remember, both DBMS and DSS overlap into some capabilities of IRS, MIS and QAS. We may want to pull some useful functions from these other system models, such as MIS's ability to make summaries based on referenced entries, QAS's attempt to communicate with and understand natural language and IRS's ability to make approximate queries. Therefore, it seems that what we really want is a synthesis of both the Data Base Management System and the Decision Support System models.

Now that a decision has been made on which system model is consistent with the users' specific needs, the next step is to acquire system software that follows that model.

System Software Alternatives

In implementing any system there is initially a choice of how to acquire the necessary software: design and write your own or obtain a software package authored by others. There are advantages and disadvantages to either alternative.

Designing and writing your own software can allow you to custom "fit" the system to your specifications. However, it will usually take more time to implement than an available software package.

Acquiring someone else's software may cost more and be less "fit" to your specific application, but it does, typically, take less time to implement.

An initial search of available bibliographic reference system software that could be implemented on the

Perkin-Elmer resulted in just one system: "KGS: A Computer Data Base System for Indexing Research Papers." The software cost satisfied our cost constraints; it was free through a Perkin-Elmer users' group library, Interchange. Upon perusal of the the system documentation, it became clear that, with a few exceptions, this package satisfied our requirements. It seemed that with a minimum of effort we could augment the software to fit our application.

The remainder of this report outlines the implementation of the KGS reference system. Chapter 2 provides a detailed description of the capabilities of the KGS data base. Chapter 3 provides a view of the logical system design. A description of the details of necessary implementation activities is included in Chapter 4. Chapter 5 covers the results of the implementation, including successful as well as unsuccessful endeavors. Finally, possible related future work is outlined in Chapter 6. An appendix includes a complete listing of source code for the KGS system, as well as a users' guide.

Chapter 2

KGS Development

The original KGS data base was developed in conjunction with the Office of Standards Development in the Occupational Safety and Health Administration (OSHA). A National Bureau of Standards project team of Lawrence Kaetzel, Dr. Robert Glass, and George Smith (hence KGS) developed the original data base software in Fortran V as a support tool for organizing documents related to an OSHA project. At a later date, Judith Calabrese made major revisions to the system design and rewrote the programs to conform to Fortran VII standards. The Calabrese revised version of the KGS data base is the object of this implementation and the subject of this report. Both versions were developed under the Perkin-Elmer OS/32 MT software operating system which permits multi-user access to the computer's hardware and software resources.

This system was designed to help users of medium and small computer systems organize, file and retrieve documents. The software allows users to access an entire data base or subsets of the data base via a computer terminal, with selected results returned to the terminal screen and optionally routed for hard-copy printout.

Following is a description of operations available for the user of KGS and a summary of the system capabilities.

Operations

Create

KGS allows the creation and filing of records which each include the following reference information: three authors, article title, publication name, publication date, reference (a physical location identifier such as "file drawer" to direct the user to the actual document), six keywords and a discipline indicator. In the APPEND mode a user is prompted for all the above information. When all information has been entered, the completed record is displayed on the terminal screen. The user may check the input data and answer the prompt "SAVE O.K.". A "YES" results in the filing of the record. A "NO" results in the discarding of the entered text.

Update

Any record stored in the KGS data base may be retrieved by user entry of the desired record's unique identification number, at which time the user is prompted for the number of the line(s) to be changed (or an "*" to end the update). When the user has completed all updates on this record, the prompt "SAVE O.K." may be answered with "YES" or "NO" resulting, respectively, in the filing of the updated version of the record or the discarding of the update attempt.

Delete

Any record stored in the data base may be deleted by

user entry of the record's unique number. The record is displayed on the terminal screen and the user is prompted with the message "DELETE THIS RECORD?". The user may respond with a "YES" or "NO".

Retrieve

The user may retrieve records from the data base according to several search criteria. Upon the prompt "ENTER EXTRACT CRITERIA (AUTHOR, KEYWORD, OR ALL)," the user may specify any of the three criteria. If the user types "AUTHOR" a record may be retrieved by a match of an entered author with any records that contain the exact same author string in any of the three author fields. If the user specifies "KEYWORD" as the criteria, another prompt of "FIELD OR STRING SEARCH BY KEYWORD?" is posed by the program. If the user chooses "FIELD" then records may be retrieved based on up to six keywords entered in the keyword field. The field search also provides the user a check for UNION (OR) and INTERSECTION (AND) conditions. If UNION (OR) is selected, the retrieved record must contain at least one of the input keywords. However, if INTERSECTION (AND) is selected, the retrieved record must contain all input keywords. If the user chooses "STRING SEARCH" the entire data base of records is searched (only within the title and keyword fields) for a match with the user input string. Finally, if the "ALL" criteria is selected by the user, a listing of all records in the data base is retrieved.

Information

KGS allows the user to request and receive information in the form of a listing, complete with number of occurrences, of all authors and/or keywords currently residing in the data base.

Summary of KGS Capabilities

When examining the KGS system with our desired system capabilities in mind, it is clear that its ease of use is marked by positive and negative features. The user is prompted with alternatives at each operation decision branch. Prompts are embedded in each operation to clue the user to each new step (prompts for choices, prompts for an information field, prompts for end of the operation session, etc.). This aspect makes the system easy for a new user since no extensive training or manuals are required in order to effectively utilize the system. Though a HELP facility is described in KGS documentation, it has been left to the implementer to provide.

One negative feature that relates to both ease of use and transparency is that the user must consistently use the same author format in record creation, update and retrieval. Each author field is seen by the system as one string and when a retrieval match is attempted, it must be an exact match with the entire name in the field (e.g., "Dr. John Smith" would not match with "John Smith" or "Dr. Smith"). KGS does, however, provide the user with transparency from the records' physical storage within the

data base.

The response time should prove to be adequate since, except for the use of "STRING SEARCH" (which sequentially searches the entire data base), all retrievals are made by use of an indexed directory look-up file structure.

Correctness of the data base is an issue that is covered well by the KGS system. All supporting dictionaries and indexes are updated at the time a record is created or updated. Entries to the dictionaries and indexes are, likewise, deleted at the time a record is deleted. To avoid the storage and search problems inherent in files marred by blank records, a file maintenance program may be triggered by the system to run regularly during non-peak periods. One problem which affects correctness is that no back-up/recovery procedures are included in the system software.

KGS software's monetary cost to the department was negligible because it was available through a free users' software library, Interchange. The software system cost is 75,000 bytes of memory while user operations are performed. The file maintenance program requires 247,750 bytes of memory. Disk storage capacity necessary to operate the KGS system is determined by the number of records in the data base. An approximate calculation to determine disk storage in bytes is 900 times the maximum number of records to be stored. This allows for storage of all files, dictionaries and indexes.

Chapter 3

This chapter gives a high-level description of the overall KGS system design by its file structure and its operations. Operations fall into two basic modes: user mode and file maintenance mode.

Data Base File Structure

The KGS system uses an indexed directory look-up file structure (see Figure 2). The data base consists of a master file, keyword dictionary, author dictionary, keyword index and author index.

The master file is unsorted and contains the details of information on each reference stored. Each record corresponds to one publication and is, therefore, identified by a unique record number. The master file record format is given in Figure 3.

Keyword and author dictionary files provide for direct retrieval of master file records without slow sequential searching of that file. The keyword dictionary contains every unique keyword that has been entered in any keyword fields of the master file. Likewise, the author dictionary is a file of all authors entered. Keyword and author dictionary records have a set of pointers that identify the record number(s) of the corresponding master file record(s) (Figure 4). Both dictionary files are sorted

Author 1 (32 characters)
Author 2 (32 characters)
Author 3 (32 characters)
Publication Date (12 characters)
Publication Title (100 characters)
Publication Name (32 characters)
Reference (20 characters)
Keyword 1 (16 characters)
Keyword 2 (16 characters)
Keyword 3 (16 characters)
Keyword 4 (16 characters)
Keyword 5 (16 characters)
Keyword 6 (16 characters)
Discipline (16 characters)

KGS Master File Record Format

-Figure 3-

Keyword Dictionary Record Format

Keyword (16 characters)
1st location on master file (4 characters)
2nd location on master file (4 characters)
.
.
.
Last Location on master file (4 characters)

Author Dictionary Record Format

Author name (32 characters)
1st location on master file (4 characters)
2nd location on master file (4 characters)
.
.
.
Last location on master file (4 characters)

Dictionary Record Formats

-Figure 4-

Keyword Index Record Format

Keyword Root (1 character)
Pointer to keyword dictionary (4 characters)

Author Index Record Format

Author Root (1 character)
Pointer to author dictionary (4 characters)

Index Record Formats

-Figure 5-

in ascending order when the file maintenance programs are run. However, between these times, an unsorted overflow area is utilized for newly added keywords or authors. It is important that the file maintenance mode is run frequently enough, recreating the dictionary files, to avoid overflow becoming large enough to cause extended search and retrieval times.

In order to decrease retrieval times, index files are provided for each dictionary. The keyword index is a sorted sequence of keyword roots (first character) and pointers that mark the first dictionary entry with that keyword root. When a keyword or author retrieval is requested, the root is obtained by a binary search (SEARCH routine) of the corresponding index file. For example, given the keyword "simulation," the root would be "s" and the index pointer would give the dictionary starting address of keywords beginning with the "s" root. The author index is handled in the same manner, but pointing into the author dictionary. Once the starting address is given, a sequential search is made from that address to the next root's beginning address within the dictionary. If the keyword (or author) has not been found, then the overflow area is searched sequentially.

Modes of Operation

User Mode

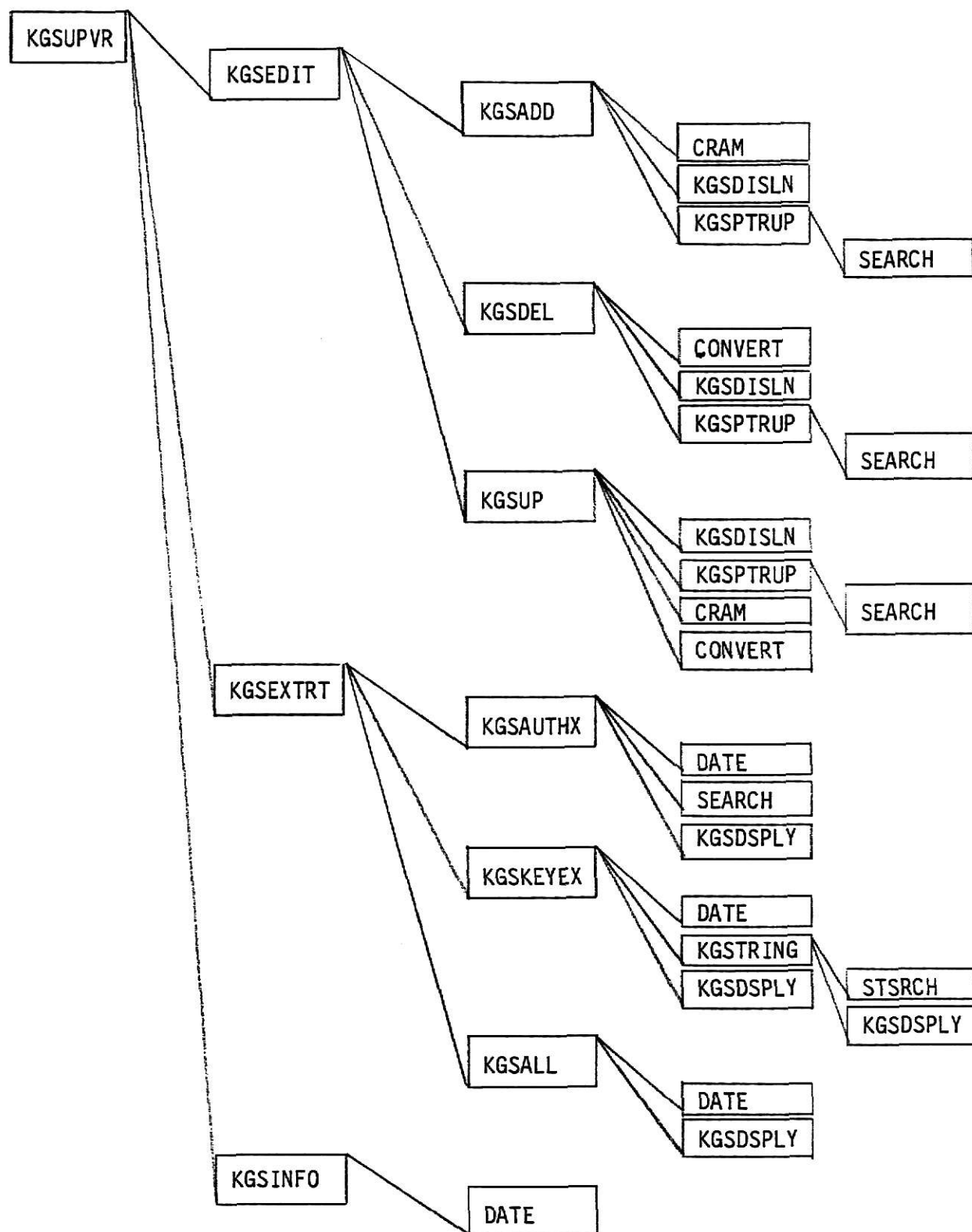
Upon user entry of "KGS", a Perkin-Elmer Command

Substitution System (CSS) command file is invoked, which loads the user mode supervisor program (KGSUPVR). This supervisor program runs all subroutines within the user mode, as diagrammed in Figure 6, which make up three user sub-modes: EDIT, EXTRACT and INFORMATION (Figure 7a).

Edit (KGSEEDIT)

Within the edit mode, there are four options: APPEND, DELETE, UPDATE and STOP. All editing retrievals require a record number within the master file. Because character entry of "*" may be made at this point (to signal the end of the editing session), user input must be read as character data. Therefore, when a record number is entered, the subroutine CONVERT right justifies the number and changes it to an integer value. The maximum number of records on the master file is limited only by disk storage capacity.

KGSADD is used when a new record is to be created and appended to the master file. While in the append mode, the system prompts the user for each element of the master file record. The CRAM routine performs the compression of two title lines into one string before storage into record format. Upon completion of record entry, the user is prompted to check the information to determine whether or not a save is desired. If the record is to be saved, author and keyword fields are simultaneously added to the dictionary files. KGSPTRUP updates the index and dictionary pointer fields.



KGS User Mode - Program Calling Structure
-Figure 6-

KGSDDEL provides the capability of deleting a record from the master file. The to-be-deleted record is displayed and the user prompted for a delete determination. If the user wants the record deleted, it is marked as a deleted node, inaccessible to users, but is not physically removed until the next file maintenance run. Author and keyword pointers to that record are deleted from the dictionary files.

KGSUP allows the user to update a record already stored in the master file. The fields of the current master file record are displayed in a 14-line format. This is accomplished with routine KGSDISLN. The user is prompted for a line number to update and for the updated value(s). The entire new string must be keyed in; no string editing facilities are available. After all updates are made, the user may save the updated version on the master file or discard it.

To exit the edit mode, the user enters "STOP", at which time control is returned to KGSUPVR (Figure 7c).

Extract (KGSEXTRT)

When information retrieval and display from the KGS data base is desired, KGSEXTRT is called. The user may choose one of three methods for retrieval: AUTHOR, KEYWORD or ALL.

KGSAUTHX allows retrieval of records by author. Index and dictionary files are utilized to locate master file records with a matching author field. As noted

KGS

*** FOR ADDITIONAL HELP OR INFORMATION
*** TYPE KGSHELP

K G S P U B L I C A T I O N S D A T A B A S E

ENTER DESIRED RUN MODE

EDIT EXTRACT INFORMATION STOP

EXTRACT

Sample KGS Output

-Figure 7a-

K G S P U B L I C A T I O N S
D A T A B A S E R E T R I E V A L M O D E

ENTER EXTRACT CRITERIA (AUTHOR, KEY, OR ALL)

KEY

FIELD OR STRING SEARCH BY KEYWORD?

FIELD

UNION (OR) OR INTERSECTION (AND)

OR

ENTER KEYWORDS

ABSOLUTE HUE

XENON FLASHTUBE

COLOUR

KEYWORD NOT IN DICTIONARY - COLOUR

ALTERNATE PRINT OPTION?

NO

KEY WORD/S SELECTED WERE: ABSOLUTE HUE XENON FLASHTUBE COLOUR

SEARCH CRITERIA USED: UNION

AUTHOR:	AKITA, M.	RECORD NO.	13
AUTHOR(2):	GRAHAM, C. H.		
AUTHOR(3):	HSIA, Y.		
FILE CODE:	0232		
PUBLICATION DATE:	0464		
TITLE:	MAINTAINING AN ABSOLUTE HUE IN PRESENCE OF DIFFERENT BACKGROUND COLORS		
PUBLICATION NAME:	VISION RES.		
REFERENCE:	4,539-556		
- - KEY WORDS - -	COLOR PERCEPTION	COLOR CONTRAST	ABSOLUTE HUE
	BACKGROUND COLOR	LUMINANCE RATIO	BEZOLD-BRUCKE

KEY WORD/S SELECTED WERE: ABSOLUTE HUE XENON FLASHTUBE COLOUR

SEARCH CRITERIA USED: UNION

AUTHOR: 9 RECORD NO. 11
AUTHOR(2): HENDLEY, C. D.
AUTHOR(3): KULIKOWSKI, J. J.
FILE CODE:
PUBLICATION DATE: 1972
TITLE: ELECTROPHYSIOLOGICAL AND PSYCHOPHYSICAL RESPONSES
TO MODULATION OF CONTRAST OF A GRATING PATTERN
PUBLICATION NAME: PERCEPTION
REFERENCE: 1,341-349
- - KEY WORDS - - XENON FLASHTUBE ELECTRONICS POWER SUPPLIES

DO YOU WISH TO CONTINUE?

NO

ENTER DESIRED RUN MODE

EDIT EXTRACT INFORMATION STOP

```

STOP
STOP
JUDY   -END OF TASK CODE=  0      CPUTIME=4.442/0.830

```

*** FOR ADDITIONAL HELP OR INFORMATION
 *** TYPE KGSHELP

SIGNOF
ELAPSED TIME=3:44 CPUTIME=4.442/0.830
TIME OFF=12/30/81 11:50:58

previously, this requires an exact match. Each record is output to the terminal screen (or printer) in a report format (by use of routine KGSDSPLY).

KGSKEYEX allows the user to specify one of two keyword search techniques: FIELD or STRING (Figures 7b-c).

If field search is requested, up to six keywords to be matched may be entered by the user. The system prompts for UNION or INTERSECTION relations between the entered keywords. If UNION (OR) is chosen, the selected record must contain at least one of the input keywords. If INTERSECTION (AND) is selected, the record must contain all input keywords.

KGSTRING allows the user to sequentially search all publication title and keyword fields within the data base. STSRCH compares the input string (maximum of 50 characters) and record strings for a match. For each match made, a record is output in report format to the terminal screen (or printer).

KGSALL sequentially reads through the author dictionary, accessing and outputting to screen and/or printer (again, in report format) all records for each author. The user receives a complete listing of the KGS master file in alphabetical order by author (with duplicated records for multi-authored entries).

Information (KGSINFO)

KGSINF allows the user to receive a listing of all dictionary entries and the number of occurrences of a

keyword or author entry within the data base. The user is prompted to choose which listings: KEYWORD, AUTHOR or BOTH (Figures 8 & 9).

File Maintenance Mode

File maintenance mode may be run by a time-sensitive CSS command file. This is quite useful for running this mode unattended during non-peak periods of system use. KGSDCNY provides maintenance by recreating dictionaries (from previous dictionary and overflow areas) and recreating indexes for each "new" dictionary. (Overflow areas are then discarded.) KGSRENEW searches for records marked for deletion, removing them from the master file. Figure 10 is a diagram of program calls within the file maintenance mode.

KGS PUBLICATIONS DATA BASE			AUTHOR LIST		12/30/81		OCCURRENCES
AUTHOR	AUTHOR	AUTHOR	OCCURRENCES	AUTHOR	AUTHOR	AUTHOR	
* 9	ABRAMOV, I.	BECK, J.	1	BECK, W. C.	BECK, W. C.	BECK, W. C.	1
	ACTKINSON, T. R.	BECK, W. C., GEFFERT, J.	1	BECK, W. C., GEFFERT, J.	BECK, W. C., GEFFERT, J.	BECK, W. C., GEFFERT, J.	14
	ADAMS, A. J.	BEDELL, H. E.	1	BEDELL, H. E.	BEDELL, H. E.	BEDELL, H. E.	1
	ADAMS, B. B.	BEDFORD, R. E.	1	BEDFORD, R. E.	BEDFORD, R. E.	BEDFORD, R. E.	1
	AKITA, M.	BEERS, J.	1	BEERS, J.	BEERS, J.	BEERS, J.	1
	AKSUGUR, E.	BEGGS, S. S.	3	BEGGS, S. S.	BEGGS, S. S.	BEGGS, S. S.	1
	ALBIN, R. E.	BEIJER, L. B.	1	BEIJER, L. B.	BEIJER, L. B.	BEIJER, L. B.	1
	ALEXANDER, J. V.	BELBIN, R. M.	1	BELBIN, R. M.	BELBIN, R. M.	BELBIN, R. M.	1
	ALLEN, C. J.	BENNETT, C. A.	1	BENNETT, C. A.	BENNETT, C. A.	BENNETT, C. A.	1
	ALLEN, C. J.	BERKOVITZ, M.	1	BERKOVITZ, M.	BERKOVITZ, M.	BERKOVITZ, M.	1
	ALLEN, F.	BERNSTEIN, A.	1	BERNSTEIN, A.	BERNSTEIN, A.	BERNSTEIN, A.	2
	ALLEN, M. J.	BERRY, R. N.	1	BERRY, R. N.	BERRY, R. N.	BERRY, R. N.	2
	ALLISON, T.	BEXTON, W. H.	2	BEXTON, W. H.	BEXTON, W. H.	BEXTON, W. H.	1
	ALPERN, M.	BICK, M. W.	2	BICK, M. W.	BICK, M. W.	BICK, M. W.	1
	AMBLER, B. A.	BIERSNER, R. J.	4	BIERSNER, R. J.	BIERSNER, R. J.	BIERSNER, R. J.	1
	AMES, A.	BINNIE, C. D.	1	BINNIE, C. D.	BINNIE, C. D.	BINNIE, C. D.	1
	ANDERSON, D. E.	BIRREN, F.	1	BIRREN, F.	BIRREN, F.	BIRREN, F.	1
	ANDERSON, E. M. S.	BIRREN, J. E.	1	BIRREN, J. E.	BIRREN, J. E.	BIRREN, J. E.	1
	ANDREWS, F. M.	BITTERMAN, M. E.	1	BITTERMAN, M. E.	BITTERMAN, M. E.	BITTERMAN, M. E.	1
	ANSI	BLACKWELL, H. R.	1	BLACKWELL, H. R.	BLACKWELL, H. R.	BLACKWELL, H. R.	7
	ANSTIS, S.	BLACKWELL, O. M.	1	BLACKWELL, O. M.	BLACKWELL, O. M.	BLACKWELL, O. M.	1
	ANSTIS, S. M.	BLAKEMORE, C.	1	BLAKEMORE, C.	BLAKEMORE, C.	BLAKEMORE, C.	1
	ARCHEA, J.	BODINGER, D. M.	1	BODINGER, D. M.	BODINGER, D. M.	BODINGER, D. M.	1
	AREND, L. E.	BODIS-WOLLNER, I.	1	BODIS-WOLLNER, I.	BODIS-WOLLNER, I.	BODIS-WOLLNER, I.	2
	ARMSTRONG, C. E.	BODMANN, H. W.	1	BODMANN, H. W.	BODMANN, H. W.	BODMANN, H. W.	1
	ARMSTRONG, R.	BONDI, K. R.	2	BONDI, K. R.	BONDI, K. R.	BONDI, K. R.	1
	ARTSCHLAGER, E.	BONEY, D. V.	1	BONEY, D. V.	BONEY, D. V.	BONEY, D. V.	1
	BAILEY, R. W.	BOOGARD, J.	2	BOOGARD, J.	BOOGARD, J.	BOOGARD, J.	2
	BAKER, C. A.	BOOHER, H. R.	1	BOOHER, H. R.	BOOHER, H. R.	BOOHER, H. R.	1
	BARLOW, H. B.	BOOKER, R. L.	1	BOOKER, R. L.	BOOKER, R. L.	BOOKER, R. L.	1
	BARNES, J. A.	BORING, E. G.	5	BORING, E. G.	BORING, E. G.	BORING, E. G.	1
	BARTLESON, C. J.	BORNSTEIN, M. H.	1	BORNSTEIN, M. H.	BORNSTEIN, M. H.	BORNSTEIN, M. H.	2
	BARTLETT, F.	BORNSTEIN, W.	2	BORNSTEIN, W.	BORNSTEIN, W.	BORNSTEIN, W.	1
	BARTLETT, N. R.	BORSELLINO, A. I.	1	BORSELLINO, A. I.	BORSELLINO, A. I.	BORSELLINO, A. I.	1
	BARTZ, A. E.	BOUMA, H.	1	BOUMA, H.	BOUMA, H.	BOUMA, H.	1
	BAUER, R. W.	BOUNAN, M. A.	1	BOUNAN, M. A.	BOUNAN, M. A.	BOUNAN, M. A.	4
	BEAPE, A. C.	BOYCE, P. R.	1	BOYCE, P. R.	BOYCE, P. R.	BOYCE, P. R.	9
	BEARE, A. N.	BOYSTON, L. E.	1	BOYSTON, L. E.	BOYSTON, L. E.	BOYSTON, L. E.	1
		BOYNTON, R. M.	1	BOYNTON, R. M.	BOYNTON, R. M.	BOYNTON, R. M.	15

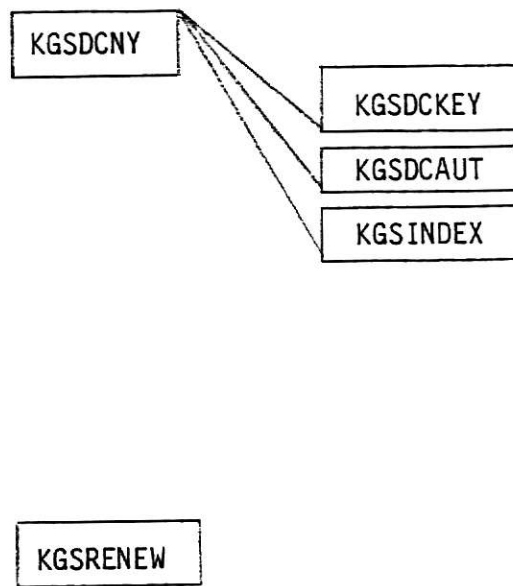
KGSINFO Sample Output for Authors

-Figure 8-

KGS PUBLICATIONS DATA BASE KEYWORD	KEYWORD LIST OCCURRENCES	12/30/81 KEYWORD	OCCURRENCES
ABSOLUTE HUE	1	BALLAST DESIGN	1
ABSORPTION SPECT	1	BAR WIDTH	1
ABSTRACTS	1	BEZOLD-BRUCKE	4
ACHROMATIC	2	BIBLIOGRAPHY	4
ACHROMATIC CHANN	1	BINOCLAR RIVALR	1
ACHROMATIC COLOR	1	BINOCLAR VISION	2
ACOUSTIC DISTORT	1	BIOLOGICAL ASPEC	1
ACOUSTICS	1	BIOLOGICAL IMPLI	1
ACTIVED	2	BIOMETEROLGY	1
ACTINIC EFFECTS	1	BLACK BAR	1
ADAPTATION	1	BLOCHS LAW	1
AGE	18	BLUE CONES	1
AGE CHANGES	7	BLUE/WHITE LIGHT	1
AGE EFFECTS	1	BLUR	4
AGING	1	BODY TEMPERATURE	1
AIR IONS	1	BORDER	4
ALPHA AMYLASE	1	BRIGHT CONSTANCY	1
ANIMAL BEHAVIOR	1	BRIGHT ENHANCE	1
ANIMAL RESEARCH	1	BRIGHT GRADIENTS	1
ANOMALOSCOPE	3	BRIGHT REVERSAL	1
ANSI	2	BRIGHTNESS	17
APERTURE	1	BRIGHTNESS CONTR	1
APPARENT BRIGHTN	1	BRIGHTNESS MATCH	1
APPARENT CONTRAS	3	BRIGHTNESS PERCE	1
APPROACH	2	BUILDINGS	1
ARTIFICIAL COLOR	1	BUSINESS DISTRIC	1
ARTIFICIAL IMAGE	1	CALIBRATION	1
ARTIFICIAL SUNLI	1	CATARACT	1
ASTIGMATISM	4	CATHODE-RAY TUBE	1
ASYMETRY	1	CEREBRAL DOMINAN	1
ASYNCHRON FLASH	1	CHLOROLABE	3
AUDITION	1	CHROMATIC	1
AUDITORY	1	CHROMATIC ADAPTA	23
AUTONOMIC NERVOU	1	CHROMATIC BORDER	2
AVERAGE LUMINANC	1	CHROMATIC CHANNE	1
AVIATION	3	CHROMATIC CONTRA	1
AVIATION SAFETY	1	CHROMATIC ILLUMI	1
BACKGROUND COLOR	1	CHROMATIC INDUCT	1
BACKWARD MASKING	2	CHROMATIC RESPON	1
		CHROMATIC ROD AC	1

KGSINFO Sample Output for Keywords

-Figure 9-



KGS File Maintenance Mode - Program Calling Structure
-Figure 10-

Chapter 4

Introduction

The implementation of the KGS data base system, as it has progressed, has involved activities directed towards a variety of system-related questions as discussed in the remainder of this chapter.

Will It Work on This Machine?

In answering this question there are four considerations: machine configuration, available operating system, available compilers and storage capacities.

Since KGS was originally implemented on the same machine as that owned by the department (a Perkin-Elmer 8/32), no configuration problems, such as data representation, were anticipated.

The original KGS was run on an OS/32 MT operating system. The same operating system was used on the KSU machine until recently when a switch to the popular UNIX operating system was made. It seemed feasible that, since other software had been converted from OS/32 MT to UNIX, that this software could also be converted.

KGS was compiled under a Fortran VII compiler. The only Fortran version available under UNIX is Fortran 77. Conversion to Fortran 77, a popular update of previous Fortran versions, was possible.

Since the KSU Perkin-Elmer has a core capacity of 750,000 bytes, the KGS maximum core requirement (under file maintenance mode) of 300,000 bytes should not be a source of trouble on this computer. At this time, the disk on which the KGS data base resides is used by other students and faculty. Since, under UNIX, no single user is restricted in the amount of disk space which they may use, it is impossible to predict the amount of free disk space available for KGS use. This means the maximum storage limit is dictated, not by the computer system, but by the disk requirements of all other disk users. Knowing the typical disk usage on the Perkin-Elmer at KSU it is clear that an initial base of 1,000 records could easily reside on the disk. As the data base grows, however, one of two alternatives may be necessitated:

- 1) The computer system operator monitors disk use, freeing space in some manner, to accommodate the growing data base.
- 2) The KGS system may need to be moved to another disk with more capacity or lower-volume users.

Can You Really Trust Documentation?

In the early stages of the implementation, it was quickly apparent that eight subroutines described in the provided documentation were not present in the provided software. Furthermore, the documentation available was not sufficiently clear in explaining the functions of these

missing routines to make rewriting the missing portions the most time-effective alternative. The search began.

Upon re-examination of the original tape from Perkin-Elmer's Interchange library, three routines were found. Their file names did not abide by the naming convention of other KGS files; they did not begin with the "KGS" root.

A phone call to the Interchange librarian was returned within a few days confirming that, indeed, the documentation referenced the routines and that the routines were missing from his files also. He volunteered to query the authors who had submitted the KGS software to Interchange about the missing routines.

Several months of letters and calls ensued until, finally, a "complete" working package arrived. On examination, it was exactly the same software with which we had started.

What happened? Four subroutines had been merged into one routine after documentation had been printed. The documentation had never been updated. The other "missing" routines turned out to be special utility programs that provided a system date and handled the carriage control and on/off switch on a printer. The authors didn't think them important enough to the basic logic of the program to note them in documentation.

Reliable documentation is a must for any software implementers or maintenance personnel. Often the software

writer is not available for questions so it is essential, as the only source of information on a software system, that documentation be clear and complete. This is a real-world problem - especially for frequently updated software. Perhaps someday we will have the facility of an integrated document-generator that reads code and interprets it into natural language documentation. Until this kind of tool can be developed and made operational, software writers must view documentation as a portion of a software system that is just as dynamic a component as the software itself and must be handled in parallel.

How Do You Convert Fortran VII to Fortran 77?

KGS was written in Fortran VII to be run on OS/32 MT. In order to convert to Fortran 77, the only Fortran compiler available on the Unix system, a conversion of source code was necessary. With no manuals available for Fortran VII, the best approach was to run the code through the Fortran 77 compiler to find potential syntactic problems. With further familiarity with the system routines and a little guesswork, some semantic conversion may be made. Fortran 77 has evolved since Fortran VII and therefore provides expanded capabilities. Another layer of conversion may be made, then, taking advantage of the more extensive facilities.

How Do You Convert CSS/MT to Shell?

It is essential to first understand the OS/32 MT environment (in which the original KGS version was written) and ways in which KGS interfaces OS/32 MT. To assist with this task, the OS/32 Operator's Reference Manual and a guide to MT may be referenced. The next step is to understand how the same software interfaces are accomplished in the UNIX environment. Some helpful references are Bourne's UNIX book (BOU82) and "An Introduction to the UNIX Shell," also by S.R. Bourne.

Operating system conversion activities involved include, but are not limited to file storage, file locking, file access authority, program handling procedures (compile, link, load), I/O interface and error handling.

What Test Procedures Should Be Used?

System testing is an extremely important phase of the software life cycle. Though KGS has reportedly been operational at another site, it is imperative that testing be done, especially because in this implementation a different operating system and compiler are used.

Good test procedures include at least unit testing and integration testing.

Unit testing is done at the module level. Some module characteristics that should be evaluated during unit testing are module interface, local data structures, "important" execution paths, error-handling paths and

boundary conditions related to the above (PRE82).

In KGS there are three main modules: KGS (user module), RENEW and DICTIONARY (file maintenance module). Though RENEW and DICTIONARY are not very complex, KGS does necessitate extensive testing of file access and update results, user-system interface problems and software-hardware problems (screen and printer options) among others. Due to this complexity, as well as the size of the KGS module, unit testing may be simplified by breaking down test units to submodules of KGSEDT, KGSEXTRT and KGSINFO (or even further to KGSADD, KGSDEL, KGSUP, KGSAUTHX, KGSKEYEX and KGSALL).

Once unit testing is completed, the integration testing of units (and sub-units) is accomplished in a bottom-up fashion. This means that the most atomic stand-alone units (or in this case, sub-modules) will be integrated and tested before integrating the next higher level of modules. The last step (at the top) will be the integration testing of the complete system with "real-life" examples.

Chapter 5

Introduction

This chapter consists of lists of implementation activities incurred during this project. Specific problems of conversion from Fortran VII to Fortran 77 are included as well as this implementer's solution. Similar listings are given for the necessary adaptations to the original KGS code which allowed the OS/32 MT-based system to run on the UNIX operating system. Additional enhancements made to the KGS system are also listed. Finally, a description of the final product is given in relation to the original design goals of this implementation.

Fortran VII to Fortran 77 Conversion

Character Type Declaration

A first attempt at compiling the original KGS source code on the Fortran 77 compiler produced pages of error listings. A conversion of the character type declarations diminished the number of errors substantially. (The integer type declaration is the same in both Fortran versions.) Fortran VII enforces a character type declaration syntax of

```
<CHARACTER*><number_of_characters><,><variable_label>
```

while Fortran 77 requires a syntax of

```
<CHARACTER><blank(s)><variable_label><*>  
                                <number_of_characters>
```

In both instances the number of characters refers to the length of a variable. For array variables a similar

conversion is made from

```
<CHARACTER><number_of_characters><,>
      <array-variable_label><(><array_size><)>
to
<CHARACTER><blank(s)><array-variable_label><*>
      <number_of_characters><(><array_size><)>
```

Examples are given below.

```
CHARACTER*100,TITLE      to  CHARACTER TITLE*100
CHARACTER*32,AUTHOR(3)   to  CHARACTER AUTHOR*32(3)
```

Six-Character Labels

Fortran 77 requires that labels must be no longer than six characters in length. For this reason many variables and subroutine names were changed (e.g., KGSDISLN to KGSDLN; LETTER1 to LET1).

Decode Routine

Fortran VII provides a system utility program, DECODE, which converts a character value to an integer value. There is no similar utility offered by Fortran 77 so a section of code was necessary to serve this function. CONVRT (see Appendix) is such a routine which, after right-justifying a character value, writes that character value to a formatted scratch file. The value is then read in integer format.

Unit Reassignment

When assigning unit numbers for external files in Fortran 77 units 5 and 6 serve special purposes. Unit 5 is seen as the standard input device (e.g., terminal keyboard) and unit 6 as the standard output device (e.g., terminal

screen). This unit assignment convention conflicted with that of Fortran VII, which sees unit 6 as a command device, handling both input and output. In many instances in the original KGS source code unit 5 was assigned for use with the KGS master file. For this reason, a consistent reassignment of external file units was necessary to avoid conflict with the Fortran enforced units 0,5 or 6 (0 is a standard error device) as well as units already assigned to other KGS files within the programs.

File Specification

Originally, file names given in OPEN statements were suitable to Fortran VII in the OS/32 MT environment (e.g., "KGS" and "IDX.DTA"). Given the UNIX directory-based file storage system, pathways were necessitated for locating KGS files for any users. The KGS files reside on directory "/usr/kgs/data" so file names were converted accordingly: "KGS" to "/usr/kgs/data/kqsmasterfile" and "IDX.DTA" to "/usr/kgs/data/keyindx". (Since lower case transactions are the preferred form in UNIX, file pathways were given in all lower case. Longer file names were also acceptable and changed for that reason.)

Inquire (SIZE=

Both versions of Fortran include an INQUIRE statement that may be used to inquire about properties of a particular named file or unit. One specifier, SIZE, used in KGS is not supported by Fortran 77. SIZE gives a value

which specifies the number of records in a file - the file size. A section of code was needed, therefore, to serve this function in KGSADD, KGSDEL and KGSUP. This piece of code is simply a read loop that counts records until the end-of-file record is detected (see Appendix for the code created to replace this routine).

Formatted Files

All reading and writing of files in KGS is accomplished with format statements. However, no specification was made in any OPEN control statements that defined files as formatted. Because KGS relies heavily on the use of direct access files and because the default form specification on direct access files in Fortran 77 is "UNFORMATTED" it was necessary to add the control statement "FORM='FORMATTED'" to each direct access file opening statement. To be consistent in this enforcement by the KGS programs, "FORM='FORMATTED'" was also added to OPEN statements for sequential access files as well.

SHARE Control

In OPEN statements in Fortran VII a control specifier of "SHARE= " is available. This allows a file's locking status to be specified as "SRO" (shared read only), "EWO" (exclusive write only), "ERW" (exclusive read write), etc. Fortran 77 does not support this facility (nor does this version of UNIX. This is an operating-system-level problem that can not be addressed in this project. For this

reason, any occurrences of SHARE were eliminated from OPEN specifier lists.

File Status "RENEW"

In addition to STATUS specifier values (in an OPEN statement) of "OLD" and "NEW", Fortran VII allows a value of "RENEW". Fortran 77 does not flag "RENEW" as an erroneous value but treats it as an unknown value. The original intent of a "RENEW" status is apparently to allow a file to be rewritten, and is used in KGS in connection with files used for dumping information to a printer - a one time occurrence, after which the file data is never used.

There are several ways in which this facility could be converted - within the program code or within a UNIX shell program. Within the program code all instances of "RENEW" could be replaced with "NEW". Fortran 77 under UNIX truncates any existing file that is opened with a status of "NEW". The alternative conversion method is with a UNIX shell program. This method requires a few lines of code just preceding shell commands to execute a program. In those few lines the "RENEW" files are removed. When an OPEN statement on one of these files is performed in the source code, a new file is created (and writing begins at the top of the file). The later alternative to this conversion was implemented primarily because the intention of file renewal is more apparent in that context. It should therefore be more easily interpreted by those programmers doing subsequent maintenance. One shell program for user mode

removes a printfile before user mode is run. Another shell program removes the key and author dictionaries and indexes, as well as a temporary copy of the master file before file maintenance mode is executed (see Appendix for the code of this routine).

Date and Carcon Routines

Two subroutines mentioned in the KGS documentation were system utility routines not available in the UNIX operating system - CARCON and DATE. CARCON was used to send special carriage control values to a printer. This was not needed in this implementation. In this conversion printing specification is performed from within a shell program. At the normal termination of a KGS session any non-empty printfiles are automatically printed.

The DATE routine provided a value for today's date which was included in several hard-copy report headings. Because hard-copy printing by UNIX provides today's date in a heading on every page, all DATE calls and IDATE variables were removed.

END and ERR

Many of the original KGS subroutines check for an end-of-file condition in order to determine if a file is initially empty or if further processing within a file is necessary. This checking is accomplished in two different ways. One way relies on the "END=N" control specifier that Fortran allows in a READ statement. If an end-of-file is

detected, control of the program is resumed at code line "N". The other way, a convention by the KGS writers, determines end-of-file in a direct-access file (which does not include a proper end-of-file record). This code reads past the end of a file and, upon detection of an input/output error of end-of-file, uses a "GO TO N" control to direct the program to line "N". This convention involves a READ statement with no "END=N" control specifier and no "ERR=M" control specifier (which would, upon detection of an error - like end-of-file - resume the program at code line "M"). Immediately following the READ statement are the two statements

```
IF (ISTAT.EQ.31.OR.ISTAT.EQ.32) GO TO N
```

```
IF (ISTAT.NE.0) GO TO M
```

(ISTAT is a variable that may be assigned the IOSTAT value from an OPEN, READ, WRITE or CLOSE operation if "IOSTAT=ISTAT" is included in the control specifier list for that operation. IOSTAT is some processor-dependent value that may be thought of as an error code for input/output performance.) These two lines of code screen end-of-file and other input/output errors.

Two problems were discovered in an attempt to run these parts of the code "as is". Though Fortran 77 is documented to support END as a control specifier, it has been determined by this writer that though the compiler is sensitive to the end-of-file condition (returning an appropriate IOSTAT value), the code for transferring of

program control to code line "N" has not been implemented in this release of Fortran 77. Furthermore, the same problem is experienced with the ERR control specifier with regards to input/output errors. Certain UNIX-specific errors are detectable with control transference effective (e.g., an illegal pathway file specification) but repeatedly input/output errors were not sensed by UNIX through Fortran 77 even though UNIX error reporting should be sensitive to input/output errors (i.e., 5 EIO i/o error (BOU83)). It is this implementer's contention that this suggests a bug in this compiler or reflects incomplete implementation of the compiler (lacking communication between 8/32 processor IOSTAT code handler and UNIX ERR handler).

In addition, upon run attempt, though end-of-file was encountered, ISTAT values of "31" or "32" failed to occur. Instead, an ISTAT value of "-1" was reported each time the end-of-file was encountered. Therefore a check for a value of "-1" was added to each KGS convention to result in the statement

```
"IF (ISTAT.EQ.-1.OR.ISTAT.EQ.31.OR.ISTAT.EQ.32) GO TO N".
```

Because of the problems with END and ERR, additional KGS-form checks for end-of-file and errors were added throughout the program source code.

The lack of proper error and end-of-file specifier support in this version of Fortran 77 represents a substantial threat to the integrity of this data base system. Input/output error handling has been improved, as

described, by the addition of end-of-file and input/output error checks. It is unknown, however, what other error problems might occur with no warning.

OS/32 MT to UNIX Conversion

Cases

All of the KGS data base system source code was written in upper case alphabetical characters. Assorted problems, mostly related to external file definition and formatting through OPEN, WRITE and READ statements, were encountered in an attempt to run upper case code in the UNIX environment. Therefore, all upper case alphabetic characters in the source code were converted to lower case alphabetic (utilizing the UNIX conversion facility "DD IF=upper_case_filename OF=lower-case-filename CONV=LCASE" (BOU83)).

Compile and Link

OS/32 MT requires CSS command programs to be used for compiling and linking Fortran VII programs. UNIX provides a nice alternative called "MAKE" which allows a programmer to compile programs, linking them into executable modules (BOU83). This facility was used to compile and link the three KGS system modules KGS, KCNY and KRENEW.

Load

Instead of using provided CSS programs for loading the KGS system modules, two UNIX shell programs were written

to load and run the two KGS modes - user and file maintenance. The KGS shell, invoked by a user typing "KGS", removes a renewable printfile, loads and runs the KGS system module and handles optional hard-copy printing. The KGSRENEW shell, invoked by a user typing "KGSRENEW" or by a pre-specified system state (probably a time-of-day). This shell program copies the existing KGS master file into a backup holding file, removes the KGS renewable temporary file, loads and runs the KRENEW module, upon successful completion changes the temporary file name to KGS master file, removes renewable files (dictionary dump, key and author dictionaries and indexes), loads and runs the KCMY module and finally, prompts the operator to remove the KGS hold file if the program terminated normally.

Enhancements

Most enhancements to the KGS system provide increased user ease. All entry by the user is to be done in lower case (as opposed to the previous all upper case format). The user is reminded of the lower case and critical formats through prompts on the screen. For example, in the append mode, the user is prompted with "enter first author - lower case only - in format: <last_name> <,>< ><first_name> example: doe, jane ". The all-lower-case format should prove easier for user entry and be less prone to retrieval matching problems inherent in upper-and-lower-case formats.

References by prompts to PUBLICATION TITLE were changed to ARTICLE TITLE since there seemed due cause for confusion between PUBLICATION TITLE and PUBLICATION NAME (referring to a journal name) in the original KGS code. Also prompts for REFERENCE were changed to REFERENCE LOCATOR to further clarify the intent of this value - a physical location of the companion reference in someone's file drawer or elsewhere.

A good error-reporting message system was included within the KGS code, especially for OPEN, WRITE, READ and CLOSE errors. Typical of these messages was "ERROR ON READING FILE - ISTAT = x" (where "x" was some IOSTAT code). One enhancement to these messages is the inclusion of the specific subroutine or program within which the message was written. This enhancement has made debugging and testing of this system much easier and should aid anyone maintaining or enhancing the system in the future.

Finally, another enhancement for any succeeding programmers is the clarification of some bits of code by additional source-code-context documentation.

KGS - The Final Product

This implementation of KGS at the Kansas State University Computer Science Department has been completed. What do we have as a result?

In an operational sense, we have a bibliographic reference system that is capable of handling editing, extracting and information gathering endeavors by a group of

users. Faculty members will be able to maintain an organized base of research materials which they may share with students or other faculty members. This is what we specified in the beginning.

To the user, the resulting product is one which should be relatively easy to use. There are sufficient clear prompts to lead any new user through the activities of adding, updating and deleting reference records. An inexperienced user should also be able to easily extract records by authors, keys or title strings, especially with the use of a general HELP facility. Listings of author and key dictionary entries or the entire data base may be made with little or no knowledge of the peculiarities of file structures so there is a high degree of data independence. Transparency is aided with clear prompts where critical value formats are needed (e.g., authors). Response time is good - less than five seconds in all but the most lengthy processes (string search of the entire data base).

Though much time was spent in understanding the KGS system, overall less time was spent in implementing this system than probably would have been spent in writing and implementing a system from "scratch". However, a totally new system design might have been able to avoid some of the peculiar compiler-operating system problems experienced in Fortran77-UNIX. A system designed in C language would seem to have been a superior alternative on this machine. Perhaps, too, a new design could have prevented some of the

problems that may plague the integrity of this data base - especially in the area of error control. The lack of file-locking capabilities by any language run on this version of UNIX also suggests potential problems for any data base that is utilized by many users.

Despite these problems, however, the KGS data base reference system presents itself as the first version of what should be a valuable tool for busy education professionals trying to organize large amounts of varied information for their use.

Chapter 6

There are still possibilities for future enhancements to the KGS data base reference system. These areas of future work are described in this chapter.

There are many improvements that could increase ease-of-use. KGS supports only very simple line-editing activities (actually line replacement). Improvements in this area would be a move to full-screen editing of special-purpose formatted screens. This could make editing procedures much simpler for the user, with less prompting for line and replacement values. This could mean greater ease of use for any level of user, but especially for one who is acquainted with full-screen editing features. A less complicated but handy improvement to the editing facilities of KGS would be the ability to do string manipulation (e.g., ch/string/string) instead of full-line replacement.

KGS is restrictive in its editing mode in that a user may only retrieve a mode for editing by submitting the appropriate record number. It might be handy to designate the record by the types of activities used in the extract mode (e.g., keywords, authors). Also, for some users, the ability to extract a reference (probably very quickly, too) by designation of the target record number could be a nice improvement.

Improvements might be made in the manner in which the user could specify keyword, author or strings for retrieval. A front-end synonym table for keywords and for

authors might help when similar meanings and spellings (or frequent misspellings) of words or authors are involved (e.g., for keywords: DATABASE, DATA BASE; for authors: WEISNER, WIESNER). It would be quite convenient to be able to specify the retrieval of all records related to a keyword root or suffix instead of picking through the system one keyword at a time (e.g., COMP* would retrieve all records with COMPUTE, COMPUTER, COMPUTING, etc. or * RETRIEVAL SYSTEMS would retrieve DATA BASE RETRIEVAL SYSTEMS, DATA RETRIEVAL SYSTEMS, FILE RETRIEVAL SYSTEMS, INFORMATION RETRIEVAL SYSTEMS, etc.).

KGS does not allow the user to retrieve using logical operators (AND, OR) on the author fields. At times it might be a valuable time saver if the user could retrieve in this manner (e.g., only references written by BEERI AND BERNSTEIN). It might also be convenient for the user if the system supported more complex combinations of AND and OR (e.g., (BEERI AND BERNSTEIN) OR RISSANEN). A future enhancement could be made that would increase this logical handling of record fields.

Another improvement to this system would be the enforcement of an author field format. This could be accomplished in a number of ways, such as breaking the author into first-, middle- and last-name fields. An adjustment to retrieval procedures would have to be made if this method were used but might logically accompany a switch to full-screen facilities with enforced cursor movement from

one name sub-field to another.

Users might want to retrieve on values other than title, author and keywords. Future work might be related to allowing retrieval by REFERENCE (e.g., get all references that are in Professor X's office, in File Cabinet n), range of PUBLICATION DATE values (e.g., published after 1980), or other fields.

KGS currently provides a hard-copy listing of a selection of references in KGS report format. One future improvement would be to give the user the choice of the current report format or a bibliography format. The bibliography format would produce references in the traditional bibliographic form, which would be ordered alphabetically by author (with the elimination of duplicate entries).

One last ease-of-use improvement could be the conversion of the general HELP facility to a context-sensitive HELP facility. This means that if the user requests "HELP" while in the update sub-mode, the HELP response would relate specifically to activities involving updating.

Response time in KGS could be improved with more complete semantic conversion to Fortran 77 or newer Fortran releases supported on UNIX. For example, more sophisticated string manipulation capabilities could eliminate the need for procedures like CRAM and CONVERT.

Since exclusive access of files is not supported in

the current version of UNIX, should a new version become available, a change to exclusive accessing of all or portions of the KGS data base would provide greater integrity in this system. Greater personal-data base integrity could be maintained if there were a means of designating record-ownership within the KGS data base. Only the owner of a record would be able to execute updates and deletes on that record.

Finally, an important future improvement to this system, especially as the data base becomes larger, would be the development of back-up/recovery procedures. This could be as simple as a regular dumping of the data base onto magnetic tape once a week. It could, alternatively, be a more complex scheme that would, after a specified number of updates to the KGS data base, dump the updated data onto tape or other back-up medium.

A P P E N D I X

KGS User's Guide

KGS is a data base system which may be used for the organization of research materials (articles, papers, etc.). In order to utilize KGS, a user needs an account established on the Perkin-Elmer 8/32 mini-computer in the Kansas State University Computer Science Department. Access may be obtained through one of many terminals in the department or through a call-in line.

1. Logon Procedure

UNIX provides a prompt for a user identification to which the user must enter an account identification. If this identification is correct, UNIX will prompt for a user password. If this, too, is correct, a page of messages is displayed. The user must wait for the "\$" to appear on the screen before entering any characters from the keyboard.

2. Activating the KGS Data Base

There are two modes of interaction with which the user should be familiar: user mode and file maintenance mode. Most users will only interact with KGS in user mode. This is the mode in which editing, extracting and information listing is done. User mode is activated by entering the command "kgs". File maintenance mode should only be run during times of non-peak system usage because it is a lengthy process. (There is no real need for most users to concern themselves with file maintenance because it is run automatically and needs no command by the user (though,

if it is needed, may be executed by entering the command "kgsrenew").)

All commands and data used by KGS must be in lower case letters. Whenever an entered string exceeds a system specified string length, the string is truncated to the accepted length and saved in this form (with a corresponding message to the screen) or the user is prompted to re-enter a shorter length string.

3. User Mode

Upon the "kgs" command, the message "KGS PUBLICATION DATA BASE - ENTER DESIRED RUN MODE - EDIT EXTRACT INFORMATION STOP" is displayed on the user's screen. This is a prompt for the user to enter "edit", "extract", "information" or "stop" to designate a choice of run mode to enter.

3.1. Edit Mode

Upon the command "edit", the message "KGS PUBLICATIONS DATA BASE EDIT MODE - ENTER EDIT MODE - append delete update stop" is displayed. This is a prompt for the user to enter "append", "delete", "update" or "stop" as a desired edit mode.

3.1.1. Append Mode

The command "append" results in the message "ENTER FIRST AUTHOR - LOWER CASE LETTERS ONLY - IN FORMAT: <last_name><,><blank><first_name> EXAMPLE: doe, jane - OR A BLANK LINE TO END". It is important that all authors are

entered in lower case letters only and in the format of last name immediately followed by a comma "," immediately followed by only one blank followed by the author's first name or initial. Any author name longer than 32 characters is truncated and saved. Up to three authors may be entered or a blank line entered to designate no more authors. (The second author may be entered after the prompt "SECOND AUTHOR - LOWER CASE LETTERS ONLY" and the third author after "THIRD AUTHOR - LOWER CASE LETTERS ONLY". Each should be entered in the same format as the first author.) Next KGS prompts "ENTER PUBLICATION DATE" which may be responded to with up to 12 characters. "ENTER FIRST LINE OF ARTICLE TITLE IN LOWER CASE ONLY" appears on the screen. Line 1 of the title may be up to 78 characters long and line 2 up to 35 characters. "ENTER PUBLICATION NAME" refers to a journal or similar publication name and may be a maximum of 32 characters. The prompt for reference locator specifies a loose format of <reference owner name><physical location><file number> for use in physically locating the paper copy of the recorded reference. The reference locator must be given with 20 characters or less so brevity in owner name is important - maybe just first or last name. Next up to six keywords may be entered after the prompt "ENTER UP TO 6 KEYWORDS - ONE KEYWORD PER LINE AND ALL LOWER CASE". Each keyword may be 16 characters maximum length or skipped by entry of a blank line. Again, all entry must be in lower case only. Finally, "ENTER DISCIPLINE" prompts for a 16

character string that describes the discipline under which this reference might be best represented (e.g., computer science). This is an optional piece of data which, at this time, is not referenced in any data base query of KGS.

The prepared record is displayed and the message "SAVE J.K.?" may be responded to with "yes" (the record is added to the data base) or "no" (the record is discarded). The user is prompted with "DO YOU WANT TO CONTINUE IN APPEND MODE?" A user's "yes" begins the cycle of prompts over again. A "no" returns the user to the choices within the edit mode.

3.1.2. Delete Mode

The command "delete" begins the delete mode. KGS prompts "ENTER # OF RECORD TO BE DELETED". The user may enter the number of a record which is to be deleted from the KGS data base. A record number not within the range of records on the master file triggers a "RECORD ENTERED IS PAST END-OF-FILE, LAST RECORD NUMBER IS n" where "n" is the largest record number on file. A message of "RECORD NUMBER HAS ALREADY BEEN DELETED" may appear if the record has been deleted since the last file maintenance run. If the record number entered by the user is suitable, the candidate record is displayed for viewing by the user before the message "DELETE THIS RECORD?" is seen. "yes" deletes the record and "no" does not delete the record. The user is then prompted "DO YOU WANT TO CONTINUE IN DELETE MODE?" and may then continue in delete mode or return to edit mode.

3.1.3. Update Mode

Upon the entry of "update" the update mode is entered. The system prompts "ENTER # OF RECORD TO BE UPDATED". Messages similar to those described in "Delete Mode" are seen if the entered record number is too large or is of an already-deleted record. Once the candidate update record is found it is displayed on the screen with the message "ENTER # OF LINE TO BE UPDATED OR AN * TO END". As an example, entry of "1" will cause "OLD FIRST AUTHOR IS xxxxxxxxxxxx, ENTER NEW FIRST AUTHOR (DOE, J) - LOWER CASE ONLY - IN FORMAT: <last_name><,><blank><first_name> EXAMPLE: doe, jane" to appear and the system waits for the updated first line author. At this point the "ENTER # OF LINE TO BE UPDATED OR AN * TO END" message appears again. A response of "*" will cause the entire record to be displayed in its updated form and the message "SAVE O.K.?" to appear. A "yes" saves the record in its updated form and "no" discards the update (so the record will remain in its pre-update attempt form on the master file). "DO YOU WANT TO CONTINUE IN UPDATE MODE?" appears and may be answered "yes" to remain or "no" to return to edit mode.

3.1.4. Stop

A user response of "stop" returns the system to run mode where the initial choice of editing, extracting, information listing or stopping is again offered.

3.2. Extract Mode

When extract mode is entered ("extract") the screen greets the user with "KGS PUBLICATIONS DATA BASE RETRIEVAL MODE - ENTER EXTRACT CRITERIA (AUTHOR, KEY OR ALL)". The user may now choose on which criteria their record querying will be based. "author" allows querying of record authors. "key" allows querying of record keywords. "all" provides a listing of all records in the KGS data base.

3.2.1. Author Extract Mode

The user is prompted to "ENTER AUTHOR NAME - (DOE, J) - FORMAT MUST MATCH RECORD". An exact match is necessary so care should be taken in entry of an author name (lower case and same format as used in append mode). If any records are found which match the entered author the message "ALTERNATE PRINT OPTION?" is shown. A "yes" means a hard-copy of the retrieved record(s) will be printed in addition to being displayed on the screen. On completion of author retrieval, the system automatically returns to retrieval mode.

3.2.2. Key Extract Mode

Within the key extract mode is a choice of two types of key searching - field or string - "FIELD OR STRING SEARCH BY KEYWORD?"

A user choice of "field" causes the system to ask "UNION (OR) OR INTERSECTION (AND)?". A field search means that only the keyword fields of the data base will be searched for an exact match. The union of user-entered

keywords means that a data base record could be retrieved if it is matched with any of the entered keywords. The intersection of user-entered keywords means that only a data base record that contains all of the user-entered keywords can be retrieved. Following the user's entry of "union" (or "or") or "intersection" (or "and") the user is asked for the list of keywords "ENTER KEYWORDS {LOWER CASE ONLY)". An alternate print option is also available for field-queried record copies.

If "string" search is desired, the message "ENTER KEY WORDS FOR SEARCH (LOWER CASE ONLY)" may be answered with a string of up to 16 characters. A serial search of the entire data base is done, checking all title and keyword fields for a match. Alternate print option is available.

Completion of either field or string search results in a return to retrieval mode.

3.2.3. All Extract Mode

A user response of "all" results in a hard-copy of the entire KGS data base. Note that there is no copy shown on the terminal screen. This copy of all data base records is displayed on the page in a report format similar to that seen in Figures 7b-c.

Upon completion of any sub-mode within the retrieval mode, the user is prompted with "DO YOU WISH TO CONTINUE?". A "yes" allows a reselection of retrieval sub-mode. A "no" returns to run mode.

3.3. Information Mode

After entry to information mode the screen will show "KGS INFORMATION MODE - ENTER INFORMATION DESIRED - AUTHOR KEY or BOTH". Reply of author designates a listing of the author dictionary. A listing of the keyword dictionary or both dictionaries may also be specified. The chosen dictionary is read, listing all unique author (or keyword) entries as well as the number of times each entry occurs. The listings are given in report format on hard-copy only.

3.4. Stop

The choice of "stop" while in run mode causes the KGS user mode program to be terminated. The user is returned to the UNIX environment.

4. Logoff Procedure

In order to logoff the 8/32 UNIX system, the user must specify either "login" or press the "control" key followed by the "d" key.

5. KGS Error Messages

There are many messages prompting the user for information, re-entry of information or system choices. It is important to note that messages of "ENTRY TOO LONG - TRUNCATED" or "ENTRY TOO LONG - RE-ENTER" are not considered KGS system error messages. KGS system error messages are marked with a "k"-prefaced message that includes an abbreviated subroutine title (e.g., "KINFO: ERROR ON READING FILE - IOSTAT = 115). (This message denotes the subroutine

in which the program was running when the error occurred, so serves as a valuable de-bugging aid.) Should the user encounter such a message, they should terminate their use of KGS and contact the 8/32 system operator to report the error message.

6. File Maintenance Mode

File maintenance mode includes two main processes. The KRENEW program looks for delete-marked records in the master file, discarding them in an updated form of the master file. The following KCNY program updates all dictionaries and indexes from the new form of the master file. After this mode has been executed, a file "KGS HOLD" holds a copy of the pre-delete process. (This file could be used to reconstruct the master file in case of a system crash during processing of KRENEW or KCNY.) The file "KGS DICT DUMP" contains a complete listing of the keywords and authors of the updated dictionaries.

7. Array Limitations

As the data base of references increases there may be need to change the present KGS size specifications for internal arrays. Though external file space may be seemingly limitless, some limitations must be specified within a Fortran program on data arrays used, but these may be changed as needed. The file maintenance programs presently specify a limit of 800 entries in the keyword dictionary, 1000 entries in the author dictionary. Also

keyword dictionaries are limited to 60 master file pointer fields and author dictionaries are limited to 50 master file pointer fields. A change to the number of pointer fields would necessitate the respecifying of a longer file length in every instance of the keyword and author dictionaries within the KGS source code.

```

c
c
c
c      k g s a d d
c
c
c
c
c
c
c
c      subroutine kgsadd(istat)
c      written by judith calabrese - 25 sep 81
c      fortran77/unix conversion by kathy miller - dec 83
c
c      provides the append mode of the kgs retrieval system.
c
c      subroutine called from kgsedit
c
c          ***** logical unit assignment *****
c
c      lu1 - kgs author index file (sequential)
c      lu2 - kgs author dictionary file (direct-opened in kgsptr)
c      lu5 - standard input device
c      lu6 - standard output device
c      lu7 - kgs keyword index file (sequential)
c      lu8 - kgs keyword dictionary file (direct-opened in kgsptr)
c      lu9 - kgs data base master file (direct)
c
c
c
c          kgs data base data elements
c
c      character author*32(3)
c      character pubdat*12
c      character title*100
c      character pubnam*32
c      character refer*20
c      character keywrd*16(6)
c      character discpn*16
c
c      character buffer*4
c      character title1*78
c      character title2*35
c      character autidx(30)
c      character wrdidx(30)
c      character spaces*2
c      character astrix
c      character iresp*4
c
c      integer*2 autptr(30), wrdptr(30), authot
c      integer*2 wrdct, irec, krec
c      integer*2 iline, itype
c
c
c          initialize values
c
c      spaces=' '
c      astrix='*'

```

```

authct=0
wrct=0

c
c      open kgs index files - author & keyword
c
c      open (1,iostat=istat,err=8010,file='/usr/kgs/data/authidx',
-      recl=5,form='formatted')
c      open (7,iostat=istat,err=8010,file='/usr/kgs/data/keyidx',
-      recl=5,form='formatted')

c
c      build index arrays
c      for dictionary search
c
c      do 2 i=1,30
c      read (1,9180,iostat=istat,end=3) autidx(i),autptr(i)
c
c      check for end-of-file condition
c      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 3
c
c      authct=authct+1
2      continue
3      do 5 i=1,30
c      read (7,9180,iostat=istat,end=10) wrdidx(i),wrdptr(i)
c
c      check for end-of-file condition
c      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 10
c
c      wrct=wrct+1
5      continue
c
c      close indexes
c
10     close (1)
c      close (7)
c
c
c      open kgs data base
c
c      open (9,iostat=istat,err=8010,file='/usr/kgs/data/kgsmaster',
-      recl=372,access='direct',form='formatted')
c
c      find end of file
c
c      irec=1
15     read(9,9005,iostat=istat,rec=irec) author(1)
c
c      check for end-of-file condition
c      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 17
c
c      irec=irec+1
c      go to 15
17     continue
c      irec=irec-1
c
c      ***** input mode *****
c
c

```

```

c          set input field = spaces
c
20  do 30 i=1,3
    author(i)='
30  continue
    pubdat='
    title1='
    title2='
    pubnam='
    refer='
    do 40 i=1,6
    keywrđ(i)='
40  continue
    discpn='

c
c
c          authors
c
100  iline=1
    write (6,9000,err=8000)
    read (5,9010,err=8000) author(1),buffer
    if (author(1).eq.spaces.or.author(1).eq.astrix) go to 7000
    if (buffer.ne.spaces) go to 8020

c
120  iline=2
    write (6,9020,err=8000)
    read (5,9010,err=8000) author(2),buffer
    if (author(2).eq.spaces) go to 200
    if (author(2).eq.astrix) go to 7000
    if (buffer.ne.spaces) go to 8020

c
140  iline=3
    write (6,9030,err=8000)
    read (5,9010,err=8000) author(3),buffer
    if (author(3).eq.spaces) go to 200
    if (author(3).eq.astrix) go to 7000
    if (buffer.ne.spaces) go to 8020

c
c          publication date
c
200  iline=4
    write (6,9040,err=8000)
    read (5,9050,err=8000) pubdat,buffer
    if (pubdat.eq.astrix) go to 7000
    if (buffer.ne.spaces) go to 8020

c
c          title
c
    write (6,9060,err=8000)
    read (5,9070,err=8000) title1
    if (title1.eq.astrix) go to 7000
    write (6,9065,err=8000)
    read (5,9075,err=8000) title2,buffer
    if (title2.eq.spaces) go to 350

c
c          remove extra spaces from title field

```

```

c
    call cram(title1,title2)
c
c
c        create 100 character title field
c
350  title(1:78)=title1(1:)
    title(79:)=title2(1:)
c
c        publication name
c
400  iline=5
    write (6,9080,err=8000)
    read (5,9010,err=8000) pubnam,buffer
    if (pubnam.eq.astrix) go to 7000
    if (buffer.ne.spaces) go to 8020
c
c        reference
c
500  iline=6
    write (6,9090,err=8000)
    read (5,9100,err=8000) refer.buffer
    if (refer.eq.astrix) go to 7000
    if (buffer.ne.spaces) go to 8020
c
c        key words
c
    write (6,9110,err=8000)
    do 650 i=1,6
    read (5,9120,err=8000) keywrd(i),buffer
    if (keywrd(i).eq.astrix) go to 7000
    if (keywrd(i).eq.spaces) go to 700
    if (buffer.eq.spaces) go to 650
    write (6,9140,err=8000) keywrd(i)
650  continue
c
c        discipline
c
700  iline=7
    write (6,9130,err=8000)
    read (5,9120,err=8000) discpn,buffer
    if (discpn.eq.astrix) go to 7000
    if (buffer.ne.spaces) go to 8020
c
c        display record
c
    krec=irec+1
    call kgsdln (author, pubdat, title, pubnam,
-           refer.keywrd, discpn, krec)
c
c
c        save record?
c
1000 write (6,9150,err=8000)
    read (5,9160,err=8000) iresp

```

```

        if (iresp.eq.'yes ') go to 2000
        if (iresp.eq.'no  ') go to 7000
        iline=1
        go to 8030
c
c          write record to kgs data base
c
2000  irec=irec+1
      write (9,9170.iostat=istat,err=8040,rec=irec)
      -      (author(i),i=1,3),pubdat,title,pubnam,
      -      refer,(keywrđ(j),j=1,6),discpn
c
c          *****  update kgs dictionary files  *****
c
c          itype=1
c          call kgsptr (author,keywrđ,irec,autidx,wrđidx,
c          -          autptr,wrđptr,authct,wrđct,itype)
c          if (itype.eq.9) go to 9999
c
c          continue
c
7000  write (6,9210.err=8000)
      read (5,9160.err=8000) iresp
      if (iresp.eq.'yes ') go to 20
      iline=2
      if (iresp.ne.'no  ') go to 8030
      istat=0
c
c          return
c
9999  close (9)
      return
c
c          error reporting and formatting
c
c
8000  write (6,8005,err=9999)
8005  format (/1x,'kadd:command device error'/)
      istat=1
      go to 9999
8010  write (6,8015,err=8000) istat
8015  format (/1x,'kadd:error on opening file - iostat = '.i4/)
      go to 9999
8020  write (6,8025,err=8000)
8025  format (/1x,'line too long - please enter again'/)
      go to (100,120,140,200,400,500,700),iline
8030  write (6,8035,err=8000)
8035  format (/1x,'please respond yes or no'/)
      go to (1000,7000),iline
8040  write (6,8045,err=8000) istat
8045  format (/1x,'kadd:error on writing file - iostat = '.i4/)
      go to 9999
c

```



```

c
c          i/o formatting
c
c
9000  format (/1x,'enter first author - lower case only -'/,1x,
-      '          in format: <last_name><,>< ><first_name>'//1x,
-      '          example: doe. jane'//1x,
-      'or a blank line to end'//)
9005  format (a32,a340)
9010  format (a32,a4)
9020  format (/1x,'enter second author if appropriate'//)
9030  format (/1x,'enter third author if appropriate'//)
9040  format (/1x,'enter publication date'//)
9050  format (a12,a4)
9060  format (/1x,'enter first line of article title'//,1x,
-      '          in lower case only'//)
9065  format (1x,'enter second line of title (in lower case, too)')
9070  format (a78)
9075  format (a35,a4)
9080  format (/1x,'enter publication name'//)
9090  format (/1x,'enter reference locator'//,1x,
-      '          in format:'//,1x,
-      '          <ref owner name><physical location><file number>'//)
9100  format (a20,a4)
9110  format (/1x,'enter up to 6 keywords - one keyword per line'//,1x,
-      '          and all lower case'//)
9120  format (a16,a4)
9130  format (/1x,'enter discipline')
9140  format (/1x,'too long!!! - keyword truncated to '.a16)
9150  format (1x,'save o.k.?'')
9160  format (a4)
9170  format (3a32,a12,a100,a32,a20,6a16,a16)
9180  format (a1,i4)
9210  format (1x,'do you want to continue in the append mode?')
end

```

```

c
c
c
c
c      a l l
c
c
c
c
c      subroutine kgsall(istat)
c
c      written by judith calabrese - 3 sep 81
c      fortran77/unix conversion by kathy miller - dec 83
c
c      produces listing of all records on kgs data base
c      sorted by author - duplicate records (with multiple
c      authors) will appear.
c
c      records in dictionary overflow will not be sorted.
c
c      * logical unit assignment *
c
c      lu2-kgs author dictionary file (direct)
c      lu3-print file (sequential)
c      lu5-standard input device
c      lu6-standard output device
c      lu9-kgs data base master file (direct)
c
c      character dcnnam*32
c      character inkey*16(6)
c
c      data elements read from kgs data base record
c
c      character author*32(3)
c      character pubdat*12
c      character title*100
c      character pubnam*32
c      character refer*20
c      character keywrđ*16(6)
c      character discpn*16
c
c      integer*2 recno(50),irec,iprtsw,ipage,iprtct,krec
c
c
c      initialize values
c
c      irec=1
c      iprtsw=1
c      ipage=0
c      iprtct=0
c
c
c      open author dictionary
c      open kgs data base
c      open print file
c

```

```

      open (9,iostat=istat,err=8010,file='/usr/kgs/data/kgsmaster',
-       recl=372,access='direct',form='formatted')
      open (2,iostat=istat,err=8010,file='/usr/kgs/data/authdict',
-       recl=232,access='direct',form='formatted')
      open (3,iostat=istat,err=8010,file='/usr/kgs/data/printfile',
-       recl=132)

c
c          read author dictionary
c          to determine start of
c          overflow records
c
      read (2,9000,iostat=istat,err=8020) iovflw
      irec=irec+1

c
c          read author dictionary
c
100  read (2,9010,iostat=istat,rec=irec)
-     dcnnam,(recno(i),i=1,50)
      irec=irec+1
      iprtct=0
      ipage=ipage+1

c
c          check for end-of-file
c
      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 500
      if (istat.ne.0) go to 8020

c
c          retrieve records from kgs data base
c
      do 200 i=1,50
      if (recno(i).eq.0) go to 100
      read (9,9020,iostat=istat,err=8020,rec=recno(i))
-     (author(j),j=1,3),
-     pubdat,title,pubnam,refer,(keywrd(k),k=1,6),discpn

c
c          print out record
c
      krec=recno(i)
      call kgsdsp (inkey,dcnnam,krec,
-       iandor,iprtsw,ipage,iprtct,
-       author,pubdat,title,pubnam,refer,keywrd,discpn)

c
200  continue
c
      go to 100

c
c          closeout
c
500  close (2,iostat=istat,err=8030)
      close (3,iostat=istat,err=8030)
      close (9,iostat=istat,err=8030)
      istat=0
      write (6,9030,err=8000)

c
c          return

```

```

c
9999 return
c
c          error reporting and formatting
c
8000 write (6,8005,err=9999)
8005 format (/1x,'kall:command device error'/)
      istat=1
      pause
      go to 9999
8010 write (6,8015,err=8000) istat
8015 format (/1x,'kall:error on opening file - iostat = ',i4)
      pause
      go to 9999
8020 write (6,8025,err=8000) istat
8025 format (/1x,'kall:error on reading file - iostat = ',i4)
      pause
      go to 9999
8030 write (6,8035,err=8000) istat
8035 format (/1x,'kall:error on closing file - iostat = ',i4)
      pause
      go to 9999
c
c          i/o formatting
c
9000 format (i4)
9010 format (a32,50i4)
9020 format (3a32,a12,a100,a32,a20,6a16,a16)
9030 format (/1x,'*****          kgs print file          *****'/)
      end

```

```

c
c
c
c
c      k g s a u t h x
c
c
c
c
c      subroutine kgsaut(istat)
c
c      written by judith calabrese - 31 august 1981
c      fortran77/unix conversion by kathy miller - dec 83
c
c      subroutine called by kgsextrt.
c
c      searches the author fields for a match
c      with a user-supplied author name.
c
c      * logical unit assignments *
c
c      lu1-kgs author index file (sequential)
c      lu2-kgs author dictionary file (direct)
c      lu3-print file (sequential)
c      lu5-standard input device
c      lu6-standard output device
c      lu9-kgs data base master file (direct)
c
c      character inauth*32
c      character dennam*32
c      character inkey*16(6)
c      character index(30)
c      character hldchr
c      character iresp*4
c
c      data elements read from kgs data base record
c
c      character author*32(3)
c      character pubdat*12
c      character title*100
c      character pubnam*32
c      character refer*20
c      character keywrđ*16(6)
c      character discpn*16
c
c      integer*2 recno(50),pointr(30),istat
c      integer*2 icell,iovflw,iprtsw,iprtct,idxct
c      integer*2 iovfl1.ipage,krec
c
c      initialize values
c
c      idxct=0
c      isub=0
c      iprtsw=0
c      iprtct=0
c      ipage=1
c      iandor=0

```

```

c
c
c      accept author name from user
c
c      write (6,9000,err=8000)
c      read (5,9010,err=8000) inauth
c      if (inauth.eq.'          ') go to 9999
c      hldchr=inauth
c
c      build index array from index file
c
c      open (1,iostat=istat,err=8010,file='/usr/kgs/data/authidx',
-      recl=5,form='formatted')
c      do 100 i=1,30
c      read (1,9020,iostat=istat,err=8020,end=150) index(i),pointr(i)
c
c      check for end-of-file condition
c      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 150
c      if (istat.ne.0) go to 8020
c
c      idxct=idxct+1
100  continue
150  close (1,iostat=istat,err=8030)
c
c      open dictionary
c
c      open (2,iostat=istat,err=8010,file='/usr/kgs/data/authdict',
-      access='direct',form='formatted')
c
c      find start of overflow records on dictionary
c
c      read (2,9030,iostat=istat,rec=1) iovflw
c
c      check for end-of-file condition
c      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 8060
c      if (istat.ne.0) go to 8020
c
c      binary search of index
c
c      call search (index,idxct,hldchr,icell)
c      irec=pointr(icell)
c      if (icell.gt.0) go to 200
c      go to 220
c
c      find author in dictionary
c
c      read (2,9040,iostat=istat,rec=irec)
200  -      dcnnam,(recno(i),i=1,50)
c
c      check for end-of-file condition
c      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 8040
c      if (istat.ne.0) go to 8020
c
c      irec=irec+1
c      if (dcnnam.lt.inauth) go to 200

```

```

        if (dcnnam.eq.inauth) go to 300
c
c          search dictionary overflow for author
c
220   iovfl1=iovflw
250   read (2,9040,iostat=istat,rec=iovfl1)
-     dcnnam,(recno(i),i=1,50)
c
c          check for end-of-file condition
c
        if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 8040
        if (istat.ne.0) go to 8020
c
        iovfl1=iovfl1+1
        if (dcnnam.eq.inauth) go to 300
        go to 250
c
c          close author dictionary
c          open kgs data base
c
300   close (2,iostat=istat,err=8030)
        open (9,iostat=istat,err=8010,file='/usr/kgs/data/kgsmaster',
-       recl=372.access='direct',form='formatted')
c
c          alternate print option
c
c          hard copy output - iprtsw=1
c          output only on terminal - iprtsw=0
c
350   write (6,9050,err=8000)
        read (5,9060,err=8000) iresp
        if (iresp.eq.'no ') go to 400
        if (iresp.ne.'yes ') go to 8050
        open (3,iostat=istat,err=8010,file='/usr/kgs/data/pfile')
        iprtsw=1
c
c          retrieve records from kgs data base
c
400   do 450 i=1,50
        if (recno(i).eq.0) go to 500
        read(9,9070,iostat=istat,err=8020,rec=recno(i))(author(j),j=1,3),
-       pubdat,title,pubnam,refer,(keywrd(k),k=1,6),discpn
c
c          display record
c
        krec=recno(i)
        call kgdsp(inkey,inauth,krec,
-       iandor,iprtsw,ipage,iprtct,
-       author,pubdat,title,pubnam,refer,keywrd,discpn)
450   continue
c
c          closeout
c
500   close (9,iostat=istat,err=8030)
        close (3,iostat=istat,err=8030)
        istat=0

```

```

c
c          return
c
9999 return
c
c          error reporting and formatting
c
8000 write (6,8005,err=9999)
8005 format (/1x,'command device error'/)
      go to 9999
8010 write (6,8015,err=8000) istat
8015 format (/1x,'kauthx:error on opening file - iostat = '.i4/)
      pause
      go to 9999
8020 write (6,8025,err=8000) istat
8025 format (/1x,'kauthx:error on reading file - iostat = '.i4/)
      pause
      go to 9999
8030 write (6,8035,err=8000) istat
8035 format (/1x,'kauthx:error on closing file - iostat = '.i4/)
      pause
      go to 9999
8040 write (6,8045,err=8000) inauth
8045 format (/1x,'author name not in dictionary - '.a32/)
      go to 500
8050 write (6,8055,err=8000)
8055 format (/1x,'please respond yes or no'/)
      go to 350
8060 write (6,8065,err=8000)
8065 format (/1x,'no names in author dictionary'/)
      go to 500
c
c          i/o formatting
c
9000 format (/1x,'enter authors name - (doe. j) ',
-        '- format must match record'/)
9010 format (a32)
9020 format (a1,i4)
9030 format (i4)
9040 format (a32,50i4)
9050 format (/1x,'alternate print option?'/)
9060 format (a4)
9070 format (3a32,a12,a100,a32,a20,6a16,a16)
      end

```



```

c      program kgsdeny
c
c      written by judith calabrese - 20 october 1981
c      fortran77/unix conversion by kathy miller - dec 83
c
c      main program of the kgs dictionary mode.
c
c      calls subroutines to create keyword and author dictionaries
c      and indexes.
c
c          logical unit assignments
c
c      lu3 - print file (sequential)
c      lu6 - standard output device
c
c
c          header
c
c      write (6,9000,err=8000)
c
c          open print file
c
c      open (3,iostat=istat,err=8020,file='/usr/kgs/data/kgsdictdump',
-      form='formatted')
c
c          create dictionaries
c          and indexes
c
c      call kgsdky
c      call kgsdat
c
c          closeout
c
c      9999 stop
c
c          error reporting and formatting
c
c      8000 write (6,8005,err=9999)
c      8005 format (/1x,'kdcny:command device error'//)
c          go to 9999
c      8020 write (6,8025,err=8000) istat
c      8025 format (/1x,'kdcny:error on opening file - iostat = '.i4//)
c          go to 9999
c
c          i/o formatting
c
c      9000 format (/10x,'k g s   d i c t i o n a r y   m o d e'//,
-      10x,'this routine creates both the author and'//,
-      10x,'keyword dictionaries and indexes'//,
-      10x,'it is a lengthy process and should be'//
-      10x,'run overnight'//)
c      end
c
c
c

```

```

c
c      k g s d c k e y
c
c
c
c
c
c      subroutine kgsdky
c
c      written by judith calabrese 14 july 1981
c      fortran77/unix conversion by kathy miller - dec 83
c
c      creates a sorted dictionary of keywords for the kgs data base.
c
c      >>>>  logical unit assignment  <<<<<
c
c      lu1 - kgs data base master file (direct - read only)
c      lu2 - kgs key dictionary output file (direct)
c      lu3 - print file for list of keys (opened in main)
c      lu6 - standard output device
c
c
c      character keynam*16( 800)
c      character iname*16(6)
c      character spaces*16
c      integer*2 keyrec( 800,60),holdct,irec
c
c      initialize values
c
c      spaces= '
c
c      do 20 i=1,800
c      keynam(i)=spaces
c      do 10 j=1,60
c      keyrec(i,j)=0
10      continue
20      continue
c      ict=0
c      irec=0
c
c      build array of first-master-file-record keywords
c
c      open(1,iostat=istat,err=8010,file='/usr/kgs/data/kgsmaster'.
-      recl=372,form='formatted',access='direct')
100      irec=irec+1
c      read (1,9000,iostat=istat,err=8020,rec=irec) (iname(i),i=1,6)
c
c      check for end-of-file condition
c      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 8050
c      if (istat.ne.0) go to 8020
c
c      do 120 i=1,6
c      if (iname(i).eq.spaces) go to 120
c      keynam(i)=iname(i)
c      keyrec(i,1)=irec
c      ict=ict+1
120      continue
c      if (ict.eq.0) go to 100

```

```

c
c      sort first keywords
c
      if (ict.eq.1) go to 200
      ict2=ict-1
140   iflag=0
      do 160 i=1,ict2
      if (keynam(i).le.keynam(i+1)) go to 160
      iflag=1
      iname(1)=keynam(i)
      keynam(i)=keynam(i+1)
      keynam(i+1)=iname(1)
160   continue
      if (iflag.eq.1) go to 140

c
c      read rest of master-file-records keys - sort into key array
c
200   irec=irec+1
      read (1.9000,iostat=istat,err=8020,rec=irec)(iname(i),i=1,6)

c
c      check for end-of-file condition
      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 300
      if (istat.ne.0) go to 8020

c
      do 280 i=1,6
      if (iname(i).eq.spaces) go to 280
      do 260 j=1,ict
      if (keynam(j).ne.iname(i)) go to 220

c
c      input key already on array
c
      do 210 k=1,60
      if (keyrec(j,k).ne.0) go to 210
      keyrec(j,k)=irec
      go to 280
210   continue

c
c      add input key to array - sort
c
220   if (keynam(j).lt.iname(i)) go to 260
      holdct=ict+1
240   keynam(holdct)=keynam(holdct-1)
      do 250 k=1,60
      keyrec(holdct,k)=keyrec(holdct-1,k)
250   continue
      holdct=holdct-1
      if (holdct.gt.j) go to 240
      keynam(j)=iname(i)
      keyrec(j,1)=irec
      do 258 k=2,60
      if (keyrec(j,k).eq.0) go to 270
      keyrec(j,k)=0
258   continue
      go to 270
260   continue
c

```

```

c          append input key to array
c
      keynam(ict+1)=iname(i)
      keyrec(ict+1,1)=irec
270    ict=ict+1
280    continue
c
      go to 200
c
c          write array to key dictionary
c
c
300    close (1,iostat=istat,err=8040)
      open(2,iostat=istat,err=8010,file='/usr/kgs/data/keydict',
-      recl=256,form='formatted',access='direct')
      iovfct=ict+1
      write (2,9120,err=8000,rec=1) iovfct
      do 420 i=1,ict
        write (2,9060,err=8000,rec=i+1) keynam(i),(keyrec(i,j),j=1,60)
        write (3,9080,err=8000) keynam(i)
420    continue
c
c          closeout
c
c
      call kgsidx (0)
      write (6,9100,err=8000) ict
9999   return
c
c          error reporting and formating
c
8000   write (6,8005,err=9999)
8005   format (1x,'kdky:command device error')
      go to 9999
8010   write (6,8015,err=8000)istat
8015   format (1x,'kdky:error on opening file - iostat = '.i4)
      go to 9999
8020   write (6,8025,err=8000) istat
8025   format (1x,'kdky:error on reading file - iostat = '.i4)
      go to 9999
8040   write (6,8045,err=8000) istat
8045   format (1x,'kdky:error on closing file - iostat = '.i4)
      go to 9999
8050   write (6,8055,err=8000)
8055   format(1x,'no records in kgsmaster file')
      go to 9999
c
c          i/o formating
c
9000   format (260x,6a16,16x)
9005   format(a32,a340)
9060   format (a16,60i4)
9080   format (1x,a16)
9100   format (1x,'keywords in dictionary = '.i8)
9120   format (i4)
      end
c
c

```

```

c
c      k g s d c a u t
c
c
c
c
c
c
c      subroutine kgsdat
c
c      written by judith calabrese 14 july 1981
c      fortran77/unix conversion by kathy miller - dec 83
c
c      creates a sorted dictionary of authors on the kgs data base.
c
c      >>>>  logical unit assignment  <<<<<
c
c      lu1 - kgs data base master file (direct - read only)
c      lu2 - kgs author dictionary output file (direct)
c      lu3 - print file for list of authors (opened in main)
c      lu6 - standard output device
c
c
c      character author*32(1000)
c      character inauth*32(3)
c      character spaces*32
c      integer*2 keyrec(1000,50),holdct,irec
c
c      initialize values
c
c      spaces='
c      do 20 i=1,1000
c      author(i)=spaces
c      do 10 j=1,50
c      keyrec(i,j)=0
10  continue
20  continue
c      ict=0
c      irec=0
c
c      build array of first-master-file-record authors
c
c      open(1,iostat=istat,err=8010,file='/usr/kgs/data/kgsmaster',
-      recl=372,form='formatted',access='direct')
100  irec=irec+1
c      read (1,9000,iostat=istat,err=8020,rec=irec) (inauth(i),i=1,3)
c
c      check for end-of-file condition
c      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 8050
c      if (istat.ne.0) go to 8020
c
c      do 120 i=1,3
c      if (inauth(i).eq.spaces) go to 120
c      author(i)=inauth(i)
c      keyrec(i,1)=irec
c      ict=ict+1
120  continue

```

```

        if (ict.eq.0) go to 100
c
c          sort first authors
c
        if (ict.eq.1) go to 200
        ict2=ict-1
140      iflag=0
        do 160 i=1,ict2
        if (author(i).le.author(i+1)) go to 160
        iflag=1
        inauth(1)=author(i)
        author(i)=author(i+1)
        author(i+1)=inauth(1)
160      continue
        if (iflag.eq.1) go to 140
c
c          read authors and sort into author array
c
c
200      irec=irec+1
        read (1,9000,iostat=istat,err=8020,rec=irec)(inauth(i),i=1,3)
c
c          check for end-of-file condition
c
        if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 300
        if (istat.ne.0) go to 8020
c
        do 280 i=1,3
        if (inauth(i).eq.spaces) go to 280
        if (inauth(i).eq.'deledeledeledeledeledeledele') go to 280
        do 260 j=1,ict
        if (author(j).ne.inauth(i)) go to 220
c
c          input key already on array
c
c
        do 210 k=1,50
        if (keyrec(j,k).ne.0) go to 210
        keyrec(j,k)=irec
        go to 280
210      continue
c
c          add input author to array - sort
c
c
220      if (author(j).lt.inauth(i)) go to 260
        holdct=ict+1
240      author(holdct)=author(holdct-1)
        do 250 k=1,50
        keyrec(holdct,k)=keyrec(holdct-1,k)
250      continue
        holdct=holdct-1
        if (holdct.gt.j) go to 240
        author(j)=inauth(i)
        keyrec(j,1)=irec
        do 258 k=2,50
        if (keyrec(j,k).eq.0) go to 270
        keyrec(j,k)=0
258      continue
        go to 270

```

```

260  continue
c
c          append input author to array
c
      author(ict+1)=inauth(i)
      keyrec(ict+1,1)=irec
270  ict=ict+1
280  continue
c
      go to 200
c
c          write array to author dictionary
c
c
300  close (1,iostat=istat,err=8040)
      open(2,iostat=istat,err=8010,file='/usr/kgs/data/authdict',
- recl=232,form='formatted',access='direct')
      iovfct=ict+1
      write (2,9120,err=8000,rec=1) iovfct
      write (3,9130,err=8000)
      do 420 i=1,ict
      write (2,9060,err=8000,rec=i+1) author(i),(keyrec(i,j),j=1,50)
      write (3,9080,err=8000) author(i)
420  continue
c
c          closeout
c
c
      close(3,iostat=istat,err=8040)
      call kgsidx (1)
      write (6,9100,err=8000) ict
9999  return
c
c          error reporting and formating
c
c
8000  write (6,8005,err=9999)
8005  format (1x,'kdat:eommand device error')
      go to 9999
8010  write (6,8015,err=8000)istat
8015  format (1x,'kdat:error on opening file - iostat = ',i4)
      go to 9999
8020  write (6,8025,err=8000) istat
8025  format (1x,'kdat:error on reading file - iostat = ',i4)
      go to 9999
8040  write (6,8045,err=8000) istat
8045  format (1x,'kdat:error on closing file - iostat = ',i4)
      go to 9999
8050  write (6,8055,err=8000)
8055  format (1x,'no records in master file')
c
c          i/o formating
c
c
9000  format (3a32,276x)
9060  format (a32,50i4)
9080  format (1x,a32)
9100  format (1x,'authors in dictionary = ',i8)
9120  format (i4)
9130  format (1h1//1x)

```

end

```

c
c
c
c
c
c      k g s i n d e x
c
c
c
c
c
c
c
c      subroutine kgsidx (ifile)
c
c      written by judith calabrese - 15 july 81
c      fortran77/unix conversion by kathy miller - dec 83
c
c      creates the index used to access the kgs dictionary
c
c      >>>>>  logical unit assignments  <<<<<
c
c      lu2 - kgs key or author dictionary (opened in kdkey or kdat)
c      lu4 - kgs key or author index file (each sequential)
c            (depends upon ifile parameter passed)
c      lu6 - standard output device
c
c
c      character let1
c      character let2*2
c      integer recno,irec
c
c      initialize values
c
c      recno=1
c      irec=2
c
c      open index file
c      ifile = 0 - open keyword index
c      ifile = 1 - open author index
c
c      if (ifile.eq.0)
c      -open (4,iostat=istat,err=8010,file='/usr/kgs/data/keyidx',
c      - recl=5,form='formatted')
c
c      if (ifile.eq.1)
c      -open (4,iostat=istat,err=8010,file='/usr/kgs/data/authidx',
c      - recl=5,form='formatted')
c
c      read in overflow pointer from dictionary
c
c      if (ifile.eq.0) read (2,9000,iostat=istat,err=8000,
c      -          rec=1) let1
c      if (ifile.eq.1) read (2,9010,iostat=istat,err=8000,
c      -          rec=1) let1
c
c      read in first key from dictionary
c

```



```

c      if (ifile.eq.0) read (2,9000,iostat=istat,err=8020,
-          rec=2) let1
c      if (ifile.eq.1) read (2,9010,iostat=istat,err=8020,
-          rec=2) let1
c      recno=recno+1
c      write (4,9020,iostat=istat,err=8040) let1,recno
c
c      read remaining keys and compare
c
c      10      irec=irec+1
c      if (ifile.eq.0) then
c          read (2,9000,iostat=istat,err=8020,rec=irec) let2
c
c          check for end-of-file condition
c          if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 9999
c          if (istat.ne.0) go to 8020
c
c      end if
c      if (ifile.eq.1) then
c          read (2,9010,iostat=istat,err=8020,rec=irec) let2
c
c          check for end-of-file condition
c          if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 9999
c          if (istat.ne.0) go to 8020
c
c      end if
c      recno=recno+1
c      if (let1.eq.let2) go to 10
c      write (4,9020,iostat=istat,err=8040) let2,recno
c      let1=let2
c      go to 10
9999  return
c
c>>>>>  error reporting & formating  <<<<<
c
8000  write (6,8005,err=9999)
8005  format (1x,'kidx:command device error')
c      go to 9999
8010  write (6,8015,err=8000) istat
8015  format (1x,'kidx:error on opening file - istat = '.i4)
c      go to 9999
8020  write (6,8025,err=8000) istat
8025  format (1x,'kidx:error on reading file - istat = '.i4)
c      go to 9999
8040  write (6,8045,err=8000) istat
8045  format (1x,'kidx:error on writing file - istat = '.i4)
c      go to 9999
c
c      >>>>>  i/o formatting  <<<<<
c
9000  format (a1,240x)
9010  format (a1,231x)
9020  format (a1,i4)
c      end

```

```

c
c
c
c
c
c
c
c      c o n v e r t
c
c
c
c
c      subroutine convrt (ascii,intger)
c
c      written by judith calabrese - 13 october 1981
c      fortran77/unix conversion by kathy miller - dec 83
c
c      converts ascii value to integer
c      after right justifying field
c
c      *      logical unit assignment      *
c
c      lu4 - temporary scratch file
c      lu6 - standard output device
c
c
c      character ascii*4
c      character field*4
c      character hldchr
c
c      integer*2 intger
c
c      field=ascii
c      j=1
c
c      right justify ascii field
c
c      do 20 i=4,1,-1
c      j=j+1
c      hldchr(1:1)=ascii(i:i)
c      if (hldchr.ne.' ') go to 40
c      field=' '
c      field(j:)=ascii(1:)
20  continue
c
c      convert field to integer
c
c      open scratch file and rewind
c
c      open (4,iostat=istat,err=8010,recl=4,status='scratch',
-      file='/usr/kgs/data/t',form='formatted')
c      rewind 4
c
c      write character value
c
c      write (4,9000,iostat=istat,err=8080) field
c      rewind 4

```

```

c
c          read integer value
c
c      read (4,9010,iostat=istat,err=8060) intger
c
c      close (4,iostat=istat,err=8030)
c
c
9999 return
c
c      ***** error reporting and formatting *****
c
8000 write (6,8005,err=9999)
8005 format (/1x,'kconvert:command device error'/)
      go to 9999
8010 write (6,8015,err=8000) istat
8015 format (/1x,'kconvert:error on opening file - iostat = ',i4/)
      go to 9999
8030 write (6,8035,err=8000) istat
8035 format (/1x,'kconvert:error on closing file - iostat = ',i4/)
      go to 9999
8060 write (6,8065,err=8000) istat
8065 format (/1x,'kconvert:error on reading file - iostat = ',i4/)
      go to 9999
8080 write (6,8085,err=8000) istat
8085 format (/1x,'kconvert:error on writing file - iostat = ',i4/)
      go to 9999
c
9000 format (a4)
9010 format (i4)
      end

```

```

c
c
c
c
c
c      c r a m
c
c
c
c
c
c
c
c
c
c
c      subroutine cram (title1,title2)
c
c      written by judith calabrese - 25 september 1981
c      fortran77/unix conversion by kathy miller - dec 83
c
c      this subroutine removes extra spaces from first line of the
c      title field.
c      it also joins the second line to the first.
c
c
c      arguments
c
c      title1 - first line of title field (78 characters).
c      title2 - second line of title (35 characters).
c
c
c      character title1*78
c      character title2*35
c
c      character spaces*2
c      character hold*2
c
c
c      initialize values
c
c      spaces=' '
c      ibyte=0
c
c      remove spaces from first line
c
c      do 100 i=1,77
c      ibyte=ibyte+1
c      last=ibyte+1
c      hold(1:2)=title1(ibyte:last)
c      if (hold.eq.spaces) go to 50
c      ictr=0
c      go to 100
50  j=ibyte+1
c      title1(ibyte:)=title1(j:)
c      ibyte=ibyte-1
c      last=last-1
c      ictr=ictr+1
100 continue
c
c      calculate # of trailing spaces on first line

```

```

c
    ifill=78-(ictr-1)
    if (ifill.gt.78) go to 300
    ifill2=ifill+1
    ibyte=0

c
c          add second line to first
c
c
    do 200 i=1,35
    ibyte=ibyte+1
    hold(1:1)=title2(ibyte:ibyte)
    title1(ifill:ifill)=hold(1:1)
    j=ibyte+1
    title2(ibyte:)=title2(j:)
    ifill=ifill+1
    ifill2=ifill2+1
    if (ifill.gt.78) go to 300
    ibyte=ibyte-1

c
200  continue

c
c          check for line too long
c
c
c
300  hold(1:2)=title2(23:24)
    if (hold.ne.spaces) go to 8010

c
c          return
c
c
9999 return

c
c          error reporting & formatting
c
c
8000 write (6,8005,err=9999)
8005  format (1x,'kcram:command device error')
    go to 9999
8010 write (6,8015,err=8000) title1,title2
8015  format (/1x,'title too long - truncated to',1x,a78/,1x,a22/)
    go to 9999
end

```

k g s d e l

subroutine kgsdel(istat)

written by judith calabrese - 9 october 1981
fortran77/unix conversion by kathy miller - dec 83

allows the user to delete records from
the kgs data base

note: records are marked for deletion.
this subroutine does not resequence the data base.

this subroutine is called from kgsedit

***** logical unit assignments *****

lu1 - kgs author index file (sequential)
lu2 - kgs author dictionary file (opened in kgsptr)
lu5 - standard input device
lu6 - standard output device
lu7 - kgs keyword index file (sequential)
lu8 - kgs keyword dictionary file (opened in kgsptr)
lu9 - kgs data base master file (direct)

kgs data base data elements

character author*32(3)
character pubdat*12
character title*100
character pubnam*32
character refer*20
character keywrd*16(6)
character discpn*16

character autidx(30)
character wrdidx(30)

character iresp*4

integer*2 autptr(30), wrdptr(30), authct, wrdct
integer*2 irec, itype, endrec

initialize values

authct=0

```

        wrdct=0

c
c          open kgs index files - author & keyword
c
        open (1,iostat=istat,err=8010,file='/usr/kgs/data/authidx',
-         recl=5,form='formatted')
        open (7,iostat=istat,err=8010,file='/usr/kgs/data/keyidx',
-         recl=5,form='formatted')

c
c          build index arrays
c          for dictionary search
c
        do 2 i=1,30
        read (1,9000,iostat=istat,end=3) autidx(i),autptr(i)

c
c          check for end-of-file condition
c          if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 3
c
        authct=authct+1
2       continue
3       do 5 i=1,30
        read (7,9000,iostat=istat,end=10) wrdidx(i),wrddptr(i)

c
c          check for end-of-file condition
c          if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 10
c
        wrdct=wrddct+1
5       continue

c
c          close indexes
c
10      close (1)
        close (7)

c
c          open kgs data base
c
        open (9,iostat=istat,err=8010,file='/usr/kgs/data/kgsmaster',
-         recl=372,access='direct',form='formatted')

c
c          find end of file
c
        endrec=1
15      read (9,9005,iostat=istat,rec=endrec) author(1)

c
c          check for end-of-file condition
c          if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 17
c
        endrec=endrec+1
        go to 15
17      continue
        endrec=endrec-1

c
c          ***** delete mode *****
c

```

```

c
c      accept record number from user
c
20  write (6,9010,err=8000)
    read (5,9020,err=8000) iresp
c
c      convert record number to integer value
c
c      call convrt (iresp,irec)
c
c      check for record past end-of-file
c
c      if (irec.gt.endrec) go to 8050
c
c      read kgs data base record
c
c      read (9,9030,iostat=istat,err=8020,rec=irec)
-      (author(i),i=1,3),pubdat,title,pubnam,
-      refer.(keywrd(j),j=1,6),discpn
c      if(author(1).eq.'deledeledeledeledeledeledele')go to 8040
c
c      display record
c
c      call kgsdln (author,pubdat,title,pubnam,
-      refer.keywrd,discpn,irec)
c
c      delete this record?
c
c      write (6,9040,err=8000)
c      read (5,9020,err=8000) iresp
c      if (iresp.ne.'yes ') go to 7000
c
c      update kgs dictionary files
c
c      itype=2
c      call kgsptr (author,keywrd,irec,autidx,wrddix,
-      autptr,wrddptr,authct,wrddct,itype)
c      if (itype.eq.9) go to 9999
c
c      rewrite deleted record
c
c      do 100 i=1,3
c      author(i)='deledeledeledeledeledeledele'
100  continue
c      write (9,9030,iostat=istat,err=8030,rec=irec)
-      (author(i),i=1,3),pubdat,title,pubnam,
-      refer.(keywrd(j),j=1,6),discpn
c
c      continue
c
7000 write (6,9050,err=8000)
     read (5,9020,err=8000) iresp
     if (iresp.eq.'yes ') go to 20
     if (iresp.ne.'no ') go to 8060
     istat=0
c

```



```

c
c          return
c
c
c
9999 close (9)
      return
c
c
c          error reporting and formatting
c
c
c
8000 write (6,8005,err=9999)
8005 format (/1x,'kdel:command device error'//)
      istat=1
      go to 9999
8010 write (6,8015,err=8000) istat
8015 format (/1x,'kdel:error on opening file - iostat = ',i4//)
      go to 9999
8020 write (6,8025,err=8000) istat
8025 format (/1x,'kdel:error on reading file - iostat = ',i4//)
      go to 9999
8030 write (6,8035,err=8000) istat
8035 format (/1x,'kdel:error on writing file - iostat = ',i4//)
      go to 9999
8040 write (6,8045,err=8000)
8045 format (/1x,'record has already been deleted'//)
      go to 7000
8050 write (6,8055,err=8000) endrec
8055 format (/1x,'record entered is past end-of-file',
-       1x,'last record number is ',i4//)
      go to 7000
8060 write (6,8065,err=8000)
8065 format (/1x,'please respond yes or no'//)
      go to 7000
c
c
c          i/o formatting
c
c
c
9000 format (a1,i4)
9005 format (a32,a340)
9010 format (/1x,'enter # of record to be deleted'//)
9020 format (a4)
9030 format (3a32,a12,a100,a32,a20,6a16,a16)
9040 format (/1x,'delete this record?'//)
9050 format (/1x,'do you want to continue in delete mode?'//)
      end

```

[illegible]

```

        write (6,9070,err=8000)
        write (6,9080,err=8000) keywrd(1)
        write (6,9090,err=8000) keywrd(2)
        write (6,9100,err=8000) keywrd(3)
        write (6,9110,err=8000) keywrd(4)
        write (6,9120,err=8000) keywrd(5)
        write (6,9130,err=8000) keywrd(6)
        write (6,9140,err=8000) discpn
c
c          return
c
9999  return
c
c
c          error reporting and formatting
c
c
8000  write (6,8005,err=9999)
8005  format (/1x,'kdisln:command device error'/)
      go to 9999
c
c
c          i/o formatting
c
c
9000  format (/1x,'***** record # ',i4,' *****')
9010  format (1x,'line 1 - ',a32)
9020  format (1x,'line 2 - ',a32)
9030  format (1x,'line 3 - ',a32)
9035  format (1x,'line 4 - ',a12)
9040  format (1x,'line 5 - ',a50/,
-      11x,a50)
9050  format (1x,'line 6 - ',a32)
9060  format (1x,'line 7 - ',a20)
9070  format (1x,'***** keywords *****')
9080  format (1x,'line 8 - ',a16)
9090  format (1x,'line 9 - ',a16)
9100  format (1x,'line 10 - ',a16)
9110  format (1x,'line 11 - ',a16)
9120  format (1x,'line 12 - ',a16)
9130  format (1x,'line 13 - ',a16)
9140  format (/1x,'line 14 - ',a16/)
c
c
      end

```

```

c
c
c
c
c      k g s d s p l y
c
c
c
c
c
c
c
c
c
c      subroutine kgsdsp(inkey,inauth,irec,
-          iandor,iprtsw,ipage,iprtct,
- author,pubdat,title,pubnam,refer,keywr,discpn)
c
c
c      written by judith calabrese - 20 august 1981
c      fortran77/unix conversion by kathy miller - dec 83
c
c      displays kgs record on crt and optionally prints hard copy
c
c          arguments passed
c
c      inkey  - array of keywords input by user (a16 format)
c      inauth - author's name input by user
c      irec  - record number on kgs data base
c
c      iandor - switch signifying union or intersection
c              0=intersection
c              1=union
c              2=string search
c      iprtsw - switch indicating optional print
c              1=hard copy
c
c      ipage  - page counter
c      iprtct - record counter
c
c      ***   kgs data base fields   ***
c      author
c      pubdat
c      title
c      pubnam
c      refer
c      keywr
c      discpn
c
c          logical unit assignments
c
c      lu3-printer file (optional - already opened)
c      lu6-standard output device
c
c      character inauth*32
c      character inkey*16(6)
c      character iextrt*12
c      character title1*50(2)
c
c          data elements from kgs record

```

```

c      character author*32(3)
      character pubdat*12
      character title*100
      character pubnam*32
      character refer*20
      character keywrd*16(6)
      character discpn*16

c
c      integer*2 irec,iprtsw,ipage,iprtct

c
c      initialize values
c
c      if (iandor.eq.0) iextrt='intersection'
      if (iandor.eq.1) iextrt='union'
      if (iandor.eq.2) iextrt='string srch'

c
c      divide title field
c
c      title1(1)(1:)=title(1:50)
      title1(2)(1:)=title(51:100)

c
c      check print switch for optional hard copy
c
c      if (iprtsw.ne.1) go to 200

c
c      headings output to printer
c
c      if (iprtct.gt.0) go to 50
      write (3,9000,err=8000)
      write (3,9010,err=8000)
      if (inauth.ne.'') then
        write (3,9020,err=8000) inauth,ipage
      else
        write (3,9030,err=8000) (inkey(i),i=1,3),ipage,
-      (inkey(j),j=4,6)
        write (3,9040,err=8000) iextrt
      end if
      write (3,9010,err=8000)

c
c      records output to printer
c
c
50  write (3,9050,err=8000) author(1),irec
      write (3,9060,err=8000) author(2)
      write (3,9070,err=8000) author(3)
      write (3,9140,err=8000) discpn
      write (3,9080,err=8000) pubdat
      write (3,9090,err=8000) (title1(i),i=1,2)
      write (3,9110,err=8000) pubnam,refer
      write (3,9120,err=8000) (keywrd(i),i=1,3)
      write (3,9130,err=8000) (keywrd(i),i=4,6)
      write (3,9010,err=8000)

c
c      records output to crt
c

```

```

200   write (6,9010,err=8000)
      if (inauth.ne.'') then
         write (6,9020,err=8000) inauth,ipage
      else
         write (6,9030,err=8000) (inkey(i),i=1,3),ipage,
-         (inkey(j),j=4,6)
         write (6,9040,err=8000) iextrt
      end if
      write (6,9010,err=8000)
      write (6,9160,err=8000) author(1),irec
      write (6,9060,err=8000) author(2)
      write (6,9070,err=8000) author(3)
      write (6,9140,err=8000) discpn
      write (6,9080,err=8000) pubdat
      write (6,9090,err=8000) (title1(i),i=1,2)
      write (6,9170,err=8000) pubnam
      write (6,9180,err=8000) refer
      write (6,9120,err=8000) (keywrd(i),i=1,3)
      write (6,9130,err=8000) (keywrd(i),i=4,6)
      write (6,9010,err=8000)

c
c      increment and set counters
c
      iprtct=iprtct+1
      if (iprtct.ne.3) go to 9999
      iprtct=0
      ipage=ipage+1

c
c      closeout
c
9999  return

c
c      error reporting and formatting
c
8000  write (6,8005,err=9999)
8005  format (1x,'kdsply:command device error')
      go to 9999

c
c      i/o formatting
c
9000  format (1h1,1x///20x,'k g s   p u b l i c a t i o n s ',1x,
- ' d a t a b a s e   r e t r i e v a l ')
9010  format (1x,'- - - - -',1x,
- '- - - - -',1x,
- '- - - - -')
9020  format (1x,'author extracted was: ',a32,1x,'page ',i4)
9030  format (1x,'key word/s selected were: ',3(a16,2x),5x,
- 'page ',i4,/28x,3(a16,2x))
9040  format (1x,'search criteria used: ',a12)
9050  format (1x,'author: ',a32,
- 10x,'record no. ',i4)
9060  format (1x,'author(2): ',a32)
9070  format (1x,'author(3): ',a32)
9080  format (1x,'publication date: ',a12)
9090  format (1x,'article title: ',a50/,22x,a50)
9110  format (1x,'publication name: ',a32,1x,

```

```
- 'ref locator: ',a20)
9120 format (1x,'- - key words - - ',3(a16,4x))
9130 format (20x,3(a16,4x))
9140 format (1x,'file code:          '.a16)
9160 format (1x,'author:            '.a32,
- 5x,'record no. ',i4)
9170 format (1x,'publication name:   '.a32)
9180 format (1x,'reference locator:  '.a20)
end
```

```

c
c
c
  subroutine kgsed(istat)
c
c  written by judith calabrese - 17 sep 81
c  fortran77/unix conversion by kathy miller - dec 83
c
c  provides 3 functions for the kgs publications data base system.
c
c      add - allows the user to enter new records.
c
c      delete - allows the user to remove existing records
c               from the kgs data base.
c
c               note: this function actually marks records
c                     for deletion rather than physically
c                     removing the record.
c
c      update - allows the user to update existing records.
c
c  in all cases the kgs dictionary is revised.
c
c      ***** logical unit assignment *****
c
c  lu6 - command device
c
c      the following logical units are
c      opened and closed within subroutines
c
c  lu1 - kgs data base
c  lu2 - kgs author index
c  lu4 - kgs author dictionary
c  lu5 - standard input device
c  lu6 - standard output device
c  lu7 - kgs keyword index
c  lu8 - kgs keyword dictionary
c
c
c  character iresp*4
c
c      header
c
c  write (6,9020,err=8000)
c
c      user enters mode of operation
c
c  100 write (6,9000,err=8000)
c      read (5,9010,err=8000) iresp
c      if (iresp.eq.'appe') go to 200
c      if (iresp.eq.'dele') go to 300
c      if (iresp.eq.'upda') go to 400
c      if (iresp.eq.'stop') go to 9999
c      if (iresp.eq.'    ') go to 9999
c      go to 8020

```



```

c
c          append
200  call kgsadd(istat)
      if (istat.ne.0) go to 9999
      go to 100

c
c          delete
c
300  call kgsdel(istat)
      if (istat.ne.0) go to 9999
      go to 100

c
c          update
c
400  call kgsup(istat)
      if (istat.ne.0) go to 9999
      go to 100

c
c          return
c
9999 return

c
c          error reporting and formatting
c
8000 write (6,8005,err=9999)
8005 format (/1x,'kedit:command device error'//)
      go to 9999
8020 write (6,8025,err=8000)
8025 format (/1x,'invalid response - please enter again'//)
      go to 100

c
c          i/o formatting
c
9000 format (/////20x,'enter edit mode - '//5x,
-      'append          delete          update          stop'////////)
9010 format (a4)
9020 format (//////////20x,'k g s   p u b l i c a t i o n s',//1x,
- 10x,'d a t a   b a s e   e d i t   m o d e'////)
      end

```

```

c      subroutine kgsext(istat)
c
c      written by judith calabrese - 21 july 1981
c      fortran77/unix conversion by kathy miller - dec 83
c
c      main program of the kgs extract mode.
c
c      provides for retrieval of kgs data base records
c      based on the following parameters:
c
c          author
c          keywords (1 to 6)
c
c      logical unit assignments
c
c      lu5-standard input device
c      lu6-standard output device
c
c      character iparam*4
c      character iresp*4
c
c          write main heading
c
c      write (6,9000,err=8000)
c
c          accept parameters from user
c
c      100 write (6,9010,err=8000)
c          read (5,9020,err=8000) iparam
c          if (iparam.eq.'auth') go to 200
c          if (iparam.eq.'key ') go to 300
c          if (iparam.eq.'all ') go to 400
c          if (iparam.eq.'   ') go to 9999
c          go to 8010
c
c          author extract
c
c      200 call kgsaut(istat)
c          if (istat.ne.0) go to 9999
c          go to 500
c
c          key extract
c
c      300 call kgskey(istat)
c          if (istat.ne.0) go to 9999
c          go to 500
c
c          all extract
c
c      400 call kgsall(istat)
c          if (istat.ne.0) go to 9999
c          go to 500

```

```

c
c          continue
c
500  write (6,9030,err=8000)
      read (5,9020,err=8000) iresp
      if (iresp.eq.'yes ') go to 100
      if (iresp.eq.'no  ') go to 9999
      go to 8020

c
c          closeout
c
9999  return
c
c          error reporting and formatting
c
8000  write (6,8005,err=9999)
8005  format (/1x,'kextrt:command device error'/)
      go to 9999
8010  write (6,8015,err=9999)
8015  format (/1x,'invalid response - please enter again'/)
      go to 100
8020  write (6,8025,err=8000)
8025  format (/1x,'please respond yes or no'/)
      go to 500

c
c          i/o formatting
c
9000  format (//////////20x,'k g s  p u b l i c a t i o n s',//1x,
- 10x,'d a t a   b a s e   r e t r i e v a l   m o d e'////)
9010  format (/1x,'enter extract criteria (author, key or all)'/)
9020  format (a4)
9030  format (/1x,'do you wish to continue?'/)
      end

```

```

c
c
c      subroutine kgsinf (istat)
c
c      written by judith calabrese - 27 october 1981
c      fortran77/unix conversion by kathy miller - dec 83
c
c      provides information on keywords and authors
c      from kgs dictionaries.
c
c      lists name and number of times a keyword or
c      author name appears in the kgs data base.
c
c          logical units
c
c      lu1 - kgs author dictionary file (direct)
c      lu2 - kgs keyword dictionary file (direct)
c      lu3 - printer file (sequential)
c      lu5 - standard input device
c      lu6 - standard output device
c
c
c      character frm*20
c      character bnk*20
c      character author*32(2)
c      character keywrđ*16(2)
c      character iresp*4
c
c      integer*2 keyrec(60),iovflw,icnt1,icnt2,irec1,irec2
c      integer*2 iboth,iline,iendsw
c
c          initialize values
c
c      iboth=0
c      iendsw=0
c
c          open print device
c
c      open (3,iostat=istat,err=8020,file='/usr/kgs/data/printfile',
-        form='formatted')
c
c          user enters mode
c
c      write (6,9000,err=8000)
20  write (6,9010,err=8000)
c      read (5,9020,err=8000) iresp
c      if (iresp.eq.' ') go to 900
c      if (iresp.eq.'key ') go to 500
c      if (iresp.eq.'auth') go to 100
c      if (iresp.eq.'both') then
c          iboth=1
c          go to 100
c      endif
c      go to 8010

```

```

c
c          ***** author mode *****
c
c          open author dictionary
c
100  open (1,iostat=istat,err=8020,file='/usr/kgs/data/authdict',
-    recl=232,access='direct',form='formatted')
    inquire(1,form=frm,blank=bnk)
c
c          find start of overflow records
c
    read (1,9030,iostat=istat,err=8030,rec=1) iovflw
    inquire(1,form=frm,blank=bnk)
    irec1=2
c
c          write report headers
c
110  write (3,9040,err=8000)
    write (3,9050,err=8000)
    write (3,9060,err=8000)
    iline=0
c
c          read first and 40th record on author dictionary
c          (40th record is read for 2-column print)
c
120  read (1,9080,iostat=istat,rec=irec1) author(1),
-    (keyrec(i),i=1,50)
c
c          check for end-of-file condition
    if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 400
    if (istat.ne.0) go to 8030
c
    irec1=irec1+1
    icnt1=0
    do 140 i=1,50
    if (keyrec(i).eq.0) go to 160
    icnt1=icnt1+1
140  continue
c
c
160  if (iendsw.eq.1) go to 200
    irec2=irec1+39
    read (1,9080,iostat=istat,rec=irec2) author(2),
-    (keyrec(i),i=1,50)
c
c          check for end-of-file condition
    if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) then
c
        iendsw=1
        go to 200
    end if
    if (istat.ne.0) go to 8030
    icnt2=0
    do 180 i=1,50
    if (keyrec(i).eq.0) go to 200
    icnt2=icnt2+1

```

```

180  continue
c
c          write out information
c
200  if (iendsw.eq.1) then
      write (3,9070,err=8000) author(1),icnt1
    else if (iendsw.eq.0) then
      write (3,9070,err=8000) author(1),icnt1,author(2),icnt2
    end if
    iline=iline+1
    if (iline.eq.40) then
      irec1=irec1+40
      go to 110
    end if
    go to 120

c
c          close author dictionary
c
c
400  close (1)
      if (iboth.eq.0) go to 900

c
c          ***** keyword mode *****
c
c          open keywrd dictionary
c
500  open (2,iostat=istat,err=8020,file='/usr/kgs/data/keydict',
-      recl=256,access='direct',form='formatted')

c
c          find start of overflow records
c
c          read (2,9030,iostat=istat,err=8030,rec=1) iovflw
      irec1=2
      iendsw=0

c
c          write report headers
c
510  write (3,9090,err=8000)
      write (3,9100,err=8000)
      write (3,9060,err=8000)
      iline=0

c
c          read first and 40th record on keywrd dictionary
c          (40th record is read for 2-column print)
c
520  read (2,9120,iostat=istat,rec=irec1) keywrd(1),
-      (keyrec(i),i=1,60)

c
c          check for end-of-file condition
      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 800
      if (istat.ne.0) go to 8030

c
      irec1=irec1+1
      icnt1=0
      do 540 i=1,60

```

```

        if (keyrec(i).eq.0) go to 560
        icnt1=icnt1+1
540    continue
c
c
560    if (iendsw.eq.1) go to 600
        irec2=irec1+39
        read (2,9120,iostat=istat,rec=irec2) keywrd(2),
-      (keyrec(i),i=1,60)
c
c          check for end-of-file condition
c
        if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) then
            iendsw=1
            go to 600
        end if
        if (istat.ne.0) go to 8030
c
        icnt2=0
        do 580 i=1,60
            if (keyrec(i).eq.0) go to 600
            icnt2=icnt2+1
580    continue
c
c          write out information
c
c
600    if (iendsw.eq.1) then
        write (3,9110,err=8000) keywrd(1),icnt1
    else if (iendsw.eq.0) then
        write (3,9110,err=8000) keywrd(1),icnt1,keywrd(2),icnt2
    end if
        iline=iline+1
        if (iline.eq.40) then
            irec1=irec1+40
            go to 510
        end if
        go to 520
c
c          close keywrd dictionary
c
c
800    close (2)
c
c
c          closeout
c
c
900    close (3)
        istat=0
c
c          return
c
c
9999   return
c
c          error reporting and formatting
c
c
8000   write (6,8005,err=8000)
8005   format (/1x,'kinfo:command device error'/)
        istat=1

```

```

      go to 9999
8010 write (6,8015,err=8000)
8015 format (/1x,'invalid response - please enter again'/)
      go to 20
8020 write (6,8025,err=8000) istat
8025 format (/1x,'kinfo:error on opening file - iostat =',i4/)
      go to 9999
8030 write (6,8035,err=8000) istat
8035 format (/1x,'kinfo:error on reading file - iostat = ',i4/)
      go to 9999
c
c          i/o formatting
c
9000 format (////////10x,'k g s   i n f o r m a t i o n   m o d e'////)
9010 format (/10x,'enter information desired',//15x,'author'.5x,
-       'key'.5x,'both'//)
9020 format (a4)
9030 format (i4)
9040 format (1h1////////,10x,'kgs publications data base',5x,
-       'author list'//)
9050 format (10x,'author'.15x,'occurrences',25x,'author'.16x,
-       'occurrences')
9060 format (1x,'-----',
-       '-----',
-       '-----'//)
9070 format (1x,a32,1x,i4,20x,a32,1x,i4)
9080 format (a32,50i4)
9090 format (1h1////////,10x,'kgs publications data base',5x,
-       'keyword list'//)
9100 format (10x,'keyword',20x,'occurrences',15x,'keyword',20x,
-       'occurrences')
9110 format (5x,a16,17x,i4,18x,a16,15x,i4)
9120 format (a16,60i4)
9130 format (a250)
      end

```



```
c
c
c
c      k g s k e y e x
c
c
c
c
c
c
c
c      subroutine kgskey(istat)
c
c      written by judith calabrese - 12 august 1981
c      forttran77/unix conversion by kathy miller - dec 83
c
c      subroutine called by kgsextrt.
c
c      provides the keyword retrieval function.
c
c          * field *
c
c      a field search searches the kgsdctny file for the
c      direct address of all kgs records containing the
c      entered keywords.  only the keyword field of the
c      data base is searched.
c      this mode also provides union(or)/intersection(and)
c      capability for record retrieval.
c
c          * string *
c
c      a string search searches the title field of all
c      records on the kgs data base for a match with
c      entered keywords.  the kgs data base is read
c      sequentially.
c
c          * logical unit assignments *
c
c      lu1-kgs key index data file (sequential)
c      lu2-kgs key dictionary file (direct)
c      lu3-print file (sequential)
c      lu4-temporary scratch file
c      lu5-standard input device
c      lu6-standard output device
c      lu9-kgs data base master file (direct if field search)
c                                   (sequential if string search)
c
c      character inkey*16(6)
c      character dcnkey*16
c      character inauth*32
c      character index(30)
c      character hldchr(6)
c      character idummy
c      character iresp*4
c
c      data elements read from kgs data base record
```

```

character author*32(3)
character pubdat*12
character title*100
character pubnam*32
character refer*20
character keywrđ*16(6)
character discpn*16

c
integer*2 recno(60),pointr(30),istat,iline,inkyct
integer*2 icell, isub, iovflw, iprtsw, iprtct, idxct
integer*2 itimes, iscrct, iovfl1, ierrct
integer*2 ipage, iscrec, ierrsw

c
c
c      initialize values
c
inkyct=0
idxct=0
isub=0
itimes=0
iscrct=0
iprtsw=0
iprtct=0
ierrct=0
iandor=0
ipage=1
inauth='
ierrsw=0
do 10 i=1,6
inkey(i)='
continue
10
c
c
c      input parameters
c      for field or string search
c
100 write (6,9000,err=8000)
read (5,9010,err=8000) iresp
if (iresp.eq.' ') go to 7000
if (iresp.eq.'fiel') go to 500
iline=1
if (iresp.ne.'stri') go to 8020

c
c
c      ***** string search *****
c
call kgstrg (istat)
go to 9999

c
c      ***** field search *****
c
c
c      union or intersection
c      parameters entered by user
c

```

```

c          intersection - iandor=0
c          union        - iandor=1
c
c
500 write (6,9020,err=8000)
    read (5,9010,err=8000) iresp
    if (iresp.eq.' ') go to 7000
    if (iresp.eq.'unio') then
        iandor=1
    elseif (iresp.eq.'or ') then
        iandor=1
    elseif (iresp.eq.'inte') then
        iandor=0
    elseif (iresp.eq.'and ') then
        iandor=0
    else
        iline=2
        go to 8020
    endif
c
c          accept key words from user
c
    write (6,9100,err=8000)
    do 520 i=1,6
        read (5,9030,err=8000) inkey(i)
        if (inkey(i).eq.' ') go to 540
        hldchr(i)=inkey(i)
        inkyct=inkyct+1
520  continue
540  if (inkyct.eq.0) go to 7000
c
c
c          build index array from index file
c
    open (1,iostat=istat,err=8010,file='/usr/kgs/data/keyidx',
-       form='formatted')
    do 560 i=1,30
        read (1,9040,iostat=istat,err=8060,end=580) index(i),pointer(i)
c        check for end-of-file condition
        if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 580
        if (istat.ne.0) go to 8060
        idxct=idxct+1
560  continue
580  close (1,iostat=istat,err=8030)
c
c          open dictionary and scratch files
c
    open (2,iostat=istat,err=8010,file='/usr/kgs/data/keydict',
-       recl=256,access='direct',form='formatted')
    open (4,iostat=istat,err=8010,recl=5,file='usr/kgs/data/t'.
-       status='scratch',form='formatted')
    rewind 4
c
c          find start of overflow records on dictionary
c
    read (2,9120,iostat=istat,rec=1) lovflw

```

```

c
c      check for end-of-file condition
c      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 8100
c      if (istat.ne.0) go to 8060
c
c      binary search of index
c
600  isub=isub+1
      if (isub.gt.inkyct) go to 900
      idummy=hldchr(isub)
      call search (index,idxct,idummy,icell)
      irec=pintr(icell)
      if (icell.gt.0) go to 700
      ierrct=ierrct+1
      go to 740
c
c      find keyword on dictionary
c
700  read (2,9050,iostat=istat,rec=irec)
      - dcnkey,(recno(i),i=1,60)
c
c      check for end-of-file condition
c      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 8070
c      if (istat.ne.0) go to 8060
c
      irec=irec+1
      if (dcnkey.lt.inkey(isub)) go to 700
      if (dcnkey.eq.inkey(isub)) go to 800
c
c
c      search overflow for keyword
c
740  iovfl1=iovflw
750  read (2,9050,iostat=istat,rec=iovfl1) dcnkey,
      - (recno(i),i=1,60)
c
c      check for end-of-file condition
c      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 8070
c      if (istat.ne.0) go to 8040
c
      iovfl1=iovfl1+1
      if (dcnkey.eq.inkey(isub)) go to 800
      go to 750
c
c      create scratch file of record numbers
c
c
c      write first record
c
800  if (iscrct.ne.0) go to 830
      itimes=1
      do 820 i=1,60
      if (recno(i).eq.0) go to 825
      write (4,9060,iostat=istat,err=8080) recno(i),itimes
      iscrct=iscrct+1

```

```

820  continue
825  rewind 4
      go to 600

c
c          write additional record numbers
c          to scratch file
c
830  do 860 i=1,60
      if (recno(i).eq.0) go to 870
840  read (4,9060,iostat=istat,err=8060,end=850) iscrec,itimes
c
c          check for end-of-file condition
      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 850
      if (istat.ne.0) go to 8060
c
      if (recno(i).ne.iscrec) go to 840
      backspace 4
      itimes=itimes+1
      write (4,9060,iostat=istat,err=8000) iscrec,itimes
      rewind 4
      go to 860
850  itimes=1
      write (4,9060,iostat=istat,err=8000) recno(i),itimes
      iscret=iscret+1
      rewind 4
860  continue
870  rewind 4
      go to 600

c
c          close dictionary file
c          open kgs data base
c          rewind scratch file
c
900  close (2,iostat=istat,err=8050)
      open (9,iostat=istat,err=8010,file='/usr/kgs/data/kgsmaster',
-      recl=372.access='direct',form='formatted')
      rewind 4

c
c          ***** record retrieval from kgs data base *****
c
c          alternate print option
c
c          hard copy output - iprtsw=1
c          output only on terminal - iprtsw=0
c
      write (6,9070,err=8000)
      read (5,9010,err=8000) iresp
      if (iresp.eq.'no ') go to 1000
      if (iresp.ne.'yes ') go to 8050
      open (3,iostat=istat,err=8010,file='/usr/kgs/data/printfile',
-      form='formatted')
      iprtsw=1

c
c          union (or)
c

```

```

1000  if (iandor.eq.0) go to 1200
1020  read (4,9060,iostat=istat,err=8060,end=7000) iscrec,itimes
c
c      check for end-of-file condition
c      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 7000
c      if (istat.ne.0) go to 8060
c
c      read (9,9080,iostat=istat,err=8060,rec=iscrec)
c      - (author(i),i=1,3),
c      - pubdat,title,pubnam,refer,(keywrd(j),j=1,6),discpn
c
c      display record
c
c      call kgsdsp(inkey,inauth,iscrec,
c      -          iandor,iprtsw,ipage,iprtct,
c      - author,pubdat,title,pubnam,refer,keywrd,discpn)
c
c      go to 1020
c
c      intersection (and)
c
c      1200  if (ierrct.ne.0) inkyct=inkyct-ierrct
c      1220  read (4,9060,iostat=istat,err=8060,end=1250) iscrec,itimes
c
c      check for end-of-file condition
c      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 1250
c      if (istat.ne.0) go to 8060
c
c      if (itimes.ne.inkyct) go to 1220
c      read(9,9080,iostat=istat,err=8060,rec=iscrec)
c      - (author(i),i=1,3),
c      - pubdat,title,pubnam,refer,(keywrd(j),j=1,6),discpn
c      ierrsw=1
c
c      display record
c
c      call kgsdsp(inkey,inauth,iscrec,
c      -          iandor,iprtsw,ipage,iprtct,
c      - author,pubdat,title,pubnam,refer,keywrd,discpn)
c      go to 1220
1250  if (ierrsw.eq.0) go to 8090
c
c      closeout
c
c      7000  close (4,iostat=istat,err=8030)
c      close (9,iostat=istat,err=8030)
c      close (3,iostat=istat,err=8030)
c      istat=0
c
c      return
c
c      9999  return
c
c      ***** error reporting and formatting *****
c
c      8000  write (6,8005,err=9999)

```

```

8005 format (/1x,'kkeyex:command device error'/)
      istat=1
      pause
      go to 9999
8010 write (6,8015,err=8000) istat
8015 format (/1x,'kkeyex:error on opening file - iostat = ',i4/)
      istat=1
      pause
      go to 9999
8020 write (6,8025,err=8000)
8025 format (/1x,'invalid response - please enter again'/)
      go to (100,500),iline
8030 write (6,8035,err=8000) istat
8035 format (/1x,'kkeyex:error on closing file - iostat = ',i4/)
      istat=1
      pause
      go to 9999
8040 write (6,8045,err=8000) inkey(isub)
8045 format (/1x,'keyword not in dictionary - ',a16/)
      rewind 2
      go to 600
8050 write (6,8055,err=8000)
8055 format (/1x,'please respond yes or no'/)
      go to 900
8060 write (6,8065,err=8000) istat
8065 format (/1x,'kkeyex:error on reading file - istat = ',i4)
      pause
      istat=1
      go to 9999
8070 write (6,8075,err=8000) inkey(isub)
8075 format (/1x,'keyword not in dictionary - ',a16)
      ierrct=ierrct+1
      go to 600
8080 write (6,8085,err=8000) istat
8085 format (/1x,'kkeyex:error on writing file - iostat = ',i4/)
      istat=1
      pause
      go to 9999
8090 write (6,8095,err=8000)
8095 format (/1x,'*** no files on data base'.
-      ' containing all selected keywords   ***'/)
      go to 7000
8100 write (6,8105,err=8000)
8105 format (/1x,'no records in dictionary'/)
      go to 7000

c
c      ***** i/o formatting *****
c
c
c
9000 format (/1x,'field or string search by keyword?')
9010 format (a4)
9020 format (/1x,'union (or) or intersection (and)')
9030 format (a16)
9040 format (a1,i4)
9050 format (a16,60i4)
9060 format (i4,i1)

```

```
9070 format (/1x,'alternate print option?')
9080 format (3a32,a12,a100,a32,a20,6a16,a16)
9100 format (/1x,'enter keywords (lower case only)')
9120 format (i4)
end
```



```

c
c
c
c
c
c      k g s p t r u p
c
c
c
c
c      subroutine kgsptr(author,keyword,irec,autidx,wrddix,
-      autptr,wrddptr,authct,wrddct,itype)
c
c      written by judith calabrese - 7 october 1981
c      fortran77/unix conversion by kathy miller - dec 83
c
c      updates pointers on kgs dictionary files
c
c      ***** logical unit assignments *****
c
c      lu2 - kgs author dictionary file (direct)
c      lu8 - kgs keyword dictionary file (direct)
c      lu6 - standard output device
c
c      arguments passed
c
c      author - read from kgs data base
c      keyword - read from kgs data base
c      irec - record number on kgs data base
c      autidx - author index (points to dictionary file)
c      wrddix - keyword index (points to dictionary file)
c      authct - count of cells on author index
c      wrddct - count of cells on keyword index
c      itype - 1 = append mode
c              2 = delete mode
c              9 = return code error
c
c
c      character author*32(3)
c      character keyword*16(6)
c
c      character autidx(30)
c      character wrddix(30)
c
c      character hldchr
c      character spaces*2
c
c      character authdc*32
c      character wrddcn*16
c
c      integer*2 autptr(30),wrddptr(30),icell,recno(60)
c      integer*2 krec,wrddct,authct,irec,ioflw.itype
c
c      initialize values

```

```

c      spaces= ' '
c
c      ***** author *****
c
c      open author dictionary
c
c      open (2,iostat=istat,err=8010,file='/usr/kgs/data/authdict',
-      recl=232,access='direct',form='formatted')
c
c      find start of overflow records in dictionary
c
c      read (2,9000,iostat=istat,rec=1) iovflw
c
c      check for end-of-file condition
c      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) then
c          iovflw=1
c          write (2,iostat=istat,err=8040,rec=1) iovflw
c      end if
c
c      search index for pointer to dictionary
c
c      do 700 i=1,3
c          if (author(i).eq.spaces) go to 700
c          hldchr(1:)=author(i)(1:1)
c          call search (autidx,authct,hldchr,icell)
c          if (icell.eq.0) go to 200
c          krec=autptr(icell)
c
c          find author in dictionary
c
c      100 read (2,9010,iostat=istat,rec=krec) authdc,(recno(j),j=1,50)
c
c          check for end-of-file condition
c          if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 300
c          if (istat.ne.0) go to 8020
c
c          krec=krec+1
c          if (authdc.lt.author(i)) go to 100
c          if (authdc.eq.author(i).and.itype.eq.1) go to 400
c          if (authdc.eq.author(i).and.itype.eq.2) go to 500
c
c          search overflow for match
c
c      200 krec=iovflw+1
c      220 read (2,9010,iostat=istat,rec=krec) authdc,(recno(j),j=1,50)
c
c          check for end-of-file condition
c          if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 300
c          if (istat.ne.0) go to 8020
c
c          krec=krec+1
c          if (authdc.ne.author(i)) go to 220
c          if (authdc.eq.author(i).and.itype.eq.1) go to 400

```

```

        if (authdc.eq.author(i).and.itype.eq.2) go to 500
c
c          author not in dictionary
c
300  recno(1)=irec
      do 350 j=2,50
        recno(j)=0
350  continue
      krec=krec+1
      go to 600

c
c          add record number to author in dictionary
c
c
400  do 450 j=1,50
      if (recno(j).ne.0) go to 450
      recno(j)=irec
      go to 600
450  continue
      icell=i
      go to 8030

c
c          delete record number from dictionary
c
c
500  do 550 j=1,50
      if (recno(j).eq.0) go to 600
      if (recno(j).ne.irec) go to 550
      do 520 k=j,50
        recno(k)=recno(k+1)
      if (recno(k).eq.0) go to 600
520  continue
550  continue

c
c          rewrite author dictionary record
c
c
600  krec=krec-1
      write(2,9010,iostat=istat,err=8040,rec=krec)
-    author(i),(recno(j),j=1,50)

c
700  continue

c
c          ***** keyword *****
c
c
c          open keyword dictionary
c
c
      open (8,iostat=istat,err=8010,file='/usr/kgs/data/keydict',
-    recl=256,access='direct',form='formatted')

c
c          find start of overflow records in dictionary
c
c
      read (8,9000,iostat=istat,rec=1) iovflw

c
c          check for end-of-file condition
      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) then
        iovflw=1
        write (8,9000,iostat=istat,err=8040,rec=1) iovflw

```

```

      end if
c
c          search index for pointer to dictionary
c
      do 1700 i=1,6
      if (keywrd(i).eq.spaces) go to 1700
      hldchr(1:)=keywrd(i)(1:1)
      call search (wrddix, wrdct, hldchr, icell)
      if (icell.eq.0) go to 1200
      krec=wrddptr(icell)
c
c          find keywrd in dictionary
c
1100  read (8,9020,iostat=istat,rec=krec) wrddcn,(recno(j),j=1,60)
c
c          check for end-of-file condition
      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 1300
      if (istat.ne.0) go to 8020
c
      krec=krec+1
      if (wrddcn.lt.keywrd(i)) go to 1100
      if (wrddcn.eq.keywrd(i).and.itype.eq.1) go to 1400
      if (wrddcn.eq.keywrd(i).and.itype.eq.2) go to 1500
c
c          search overflow for match
c
1200  krec=iovfllw+1
1220  read (8,9020,iostat=istat,rec=krec) wrddcn,(recno(j),j=1,60)
c
c          check for end-of-file condition
      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 1300
      if (istat.ne.0) go to 8020
c
      krec=krec+1
      if (wrddcn.ne.keywrd(i)) go to 1220
      if (wrddcn.eq.keywrd(i).and.itype.eq.1) go to 1400
      if (wrddcn.eq.keywrd(i).and.itype.eq.2) go to 1500
c
c          keywrd not in dictionary
c
1300  recno(1)=irec
      do 1350 j=2,60
      recno(j)=0
1350  continue
      krec=krec+1
      go to 1600
c
c          add record number to keywrd in dictionary
c
1400  do 1450 j=1,60
      if (recno(j).ne.0) go to 1450
      recno(j)=irec
      go to 1600
1450  continue
      icell=i
      go to 8050

```

```

c
c          delete record number from dictionary
c
1500 do 1560 j=1,60
      if (recno(j).eq.0) go to 1600
      if (recno(j).ne.irec) go to 1560
      do 1520 k=j,60
        recno(k)=recno(k+1)
        if (recno(k).eq.0) go to 1600
1520 continue
1560 continue
c
c          rewrite keywrd dictionary record
c
1600 krec=krec-1
      write(8,9020,iostat=istat,err=8040,rec=krec)
      -      keywrd(i),(recno(j),j=1,60)
c
1700 continue
c
c          return
c
c
9999 close (2)
      close (8)
      return
c
c          error reporting and formatting
c
8000 write (6,8005,err=9999)
8005 format (/1x,'kptrup:command device error'/)
      itype=9
      go to 9999
8010 write (6,8015,err=8000) istat
8015 format (/1x,'kptrup:error on opening file - iostat = ',i4/)
      itype=9
      go to 9999
8020 write (6,8025,err=8000) istat
8025 format (/1x,'kptrup:error on reading file - iostat = ',i4/)
      itype=9
      go to 9999
8030 write (6,8035,err=8000) author(icell)
8035 format (/1x,'kptrup:dictionary overflow on author - ',a32/)
      itype=9
      go to 9999
8040 write (6,8045,err=8000) istat
8045 format (/1x,'kptrup:error on writing file - iostat = ',i4/)
      itype=9
      go to 9999
8050 write (6,8055,err=8000) keywrd(icell)
8055 format (/1x,'kptrup:dictionary overflow on keyword - ',a16/)
      itype=9
      go to 9999
c
c          i/o formatting

```

```
c
9000 format (i4)
9010 format (a32,50i4)
9020 format (a16,60i4)
end
```

```

c      program kgsrenew
c
c      written by judith calabrese - 21 october 1981
c      fortran77/unix conversion by kathy miller - dec 83
c
c      removes deleted records from the kgs data base
c
c          logical unit assignments
c
c      lu1 - kgs data base master file (direct-read only)
c      lu2 - kgstemp - output file (direct-write only)
c      lu6 - standard output device
c
c          kgs data items
c
c      character author*32(3)
c      character pubdat*12
c      character title*100
c      character pubnam*32
c      character refer*20
c      character keywrd*16(6)
c      character discpn*16
c
c      integer*2 inct,outct,irec,orec
c
c          initialize values
c
c      inct=0
c      outct=0
c      irec=0
c      orecc=0
c
c          open files
c
c      open (1,iostat=istat,err=8010,file='usr/kgs/data/kgsmaster',
-      recl=372,form='formatted',access='direct')
c
c      open (2,iostat=istat,err=8010,file='usr/kgs/data/kgstemp',
-      recl=372,form='formatted',access='direct')
c
c          read kgs records
c
c      100  irec=irec+1
c          read (1,9000,iostat=istat,err=8020,rec=irec)
c          -      (author(i),i=1,3),pubdat,title,pubnam,refer,
c          -      (keywrd(j),j=1,6),discpn
c          check for end-of-file condition
c          if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 500
c          if (istat.eq.0) go to 8020
c          inct=inct+1
c
c          check for deleted records
c
c          if (author(1).eq.'deledeledeledeledeledeledele') go to 100
c
c          write out good record

```

```

c
    orec=orec+1
    write (2,9000,iostat=istat,err=8030,rec=orec)
-      (author(i),i=1,3),pubdat,title,pubnam,refer,
-      (keywrd(j),j=1,6),discpn
    if (istat.ne.0) go to 8030
    outct=outct+1
    go to 100

c
c          closeout
c
500  write (6,9010,err=8000) inct,outct
      close (1)
      close (2)
9999 stop

c
c          error reporting and formatting
c
8000 write (6,8005,err=9999)
8005 format (/1x,'krenew:command device error'/)
      go to 9999
8010 write (6,8015,err=8000) istat
8015 format (/1x,'krenew:error on opening file - iostat = ',i4/)
      go to 9999
8020 write (6,8025,err=8000) istat
8025 format (/1x,'krenew:error on reading file - iostat = ',i4/)
      go to 9999
8030 write (6,8035,err=8000) istat
8035 format (/1x,'krenew:error on writing file - iostat = ',i4/)
      go to 9999

c
c          i/o formatting
c
9000 format (3a32,a12,a100,a32,a20,6a16,a16)
9010 format (/1x,'input records = ',i4,/1x,'output records = ',i4/)
      end

```



```

c
c
c
c      s e a r c h
c
c
c
c
c
c
c      subroutine search(index,ictidx,idummy,i)
c
c      written by judith calabrese - 23 july 1981
c      fortran77/unix conversion by kathy miller - dec 83
c
c      performs binary search of index array
c
c      character index(30)
c      character idummy
c      integer*2 i,hi,lo,mid,ictidx
c
c      initialize values
c
c      if (ictidx.le.0) go to 9999
c      hi=ictidx
c      lo=1
c
c      calculate mid point
c
c      mid=(hi+lo)/2
c
c      search
c
c      if (idummy.eq.index(mid)) go to 50
c      if (idummy.gt.index(mid)) lo=mid
c      if (idummy.lt.index(mid)) hi=mid
c
c      check for last 2 records
c
c      if ((hi-lo).eq.1) go to 30
c      go to 10
c
c      final search
c
c      mid=hi
c      if (idummy.eq.index(mid)) go to 50
c      mid=lo
c      if (idummy.eq.index(mid)) go to 50
c
c      not on index
c
c      go to 8000
c
c      closeout
c
c      50  i=mid
c      9999 return
c

```

```
c          error reporting
c
8000 write (6,8005,err=9999) idummy
8005 format (1x,'input key or author root not on index - ',a1)
      i=0
      go to 9999
      end
```

```
subroutine kgstrg (istat)
```

fortran77/unix conversion by kathy miller - dec 83

logical unit assignments

lu3-print file (sequential)

lu5-standard input device

1u6-standard output device

lu9-kgs data base master file (direct)

```
character inauth#32
```

character inkey#16(6)

character instrg*16

character field#16

character iresp#4

data elements read from kgs data base record

character author#32(3)

character pubdat#12

character title#100

character pubnam#32

character refer#20

character keywrd#16(6)

character discpn#16

```
integer*2 inkyct,iprtsw,ipage,iprtct,irec
```

```
initialize values
```

inauth='

```
inkyct=0
```

```
irec=0
```

iandor=2

```
iprtsw=0
```

```
ipage=1
```

```
iprtct=0
```

istyp= '

alternate print option

hard copy output - iprtsw=1

output only on terminal - iprtsw=0

```

30  write (6,9030,err=8000)
    read (5,9040,err=8000) iresp
    if (iresp.eq.'no ') go to 40
    if (iresp.ne.'yes ') go to 8030
    open(3,iostat=istat,err=8010,file='/usr/kgs/data/printfile',
-      form='formatted')
    iprtsw=1
c
c      accept input keys from user
c
c
c
40  write (6,9000,err=8000)
    do 50 i=1,6
    read (5,9010,err=8000) inkey(i)
    if (inkey(i).eq.' ') go to 80
    inkyct=inkyct+1
50  continue
80  if (inkyct.eq.0) go to 900
c
c      open kgs data base
c
c      open (9,iostat=istat,err=8010,file='/usr/kgs/data/kgsmaster',
-      recl=372,form='formatted',access='direct')
c
c      read data base records
c
c
200 irec=i+1
    read (9,9020,iostat=istat,err=8020,rec=i)
-      (author(i),i=1,3),pubdat,title,
-      pubnam,refer.(keywrd(j),j=1,6),discpn
c
c      check for end-of-file condition
    if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 900
    if (istat.ne.0) go to 8020
c
c
c      check for deleted record
c
    if (author(1).eq.'deledeledeledeledeledeledeledele') go to 200
c
c
c
c
c      perform string search
c      on title field
c
    do 400 i=1,inkyct
    instrg=inkey(i)
    call stsrch (title,instrg,match)
    if (match.eq.1) go to 800
400 continue
c
c      perform string search
c      on key word fields on kgs data base
c

```

```

do 600 i=1,inkycr
instrg=inkey(i)
do 650 j=1,6
field=keywrd(j)
call stsrch (field,instrg,match)
if (match.eq.1) go to 800
650 continue
600 continue
c
c          no match
c
c          go to 200
c
c
c          match -
c          display record
c
800 call kgsdsp (inkey,inauth.irec,
- iandor,iprtsw,ipage,iprtct,
- author,pubdat,title,pubnam,refer.keywrd,discpn)
match=0
go to 200
c
c          closeout
c
900 close (9,iostat=istat,err=8030)
close (3,iostat=istat,err=8030)
istat=0
c
c          return
c
9999 return
c
c          error reporting and formatting
c
8000 write (6,8005,err=9999)
8005 format (/1x,'kstrng:command device error'/)
istat=1
go to 9999
8010 write (6,8015,err=8000) istat
8015 format (/1x,'kstrng:error on opening file - iostat = ',i4/)
go to 9999
8020 write (6,8025,err=8000) istat
8025 format (1x,'kstrng:error on reading file - iostat = ',i4)
go to 9999
8030 write (6,8035,err=8000)
8035 format (/1x,'please respond yes or no'/)
go to 30
c
c          i/o formatting
c
9000 format (/1x,'enter key words for search (lower case only)'/)
9010 format (a16)
9020 format (3a32,a12,a100,a32,a20.6a16,a16)
9030 format (/1x,'alternate print option?'/)
9040 format (a4)

```

end

[illegible]

```
c      ictr=0
      do 60 j=1,length
      k=j+1
      hold(1)(1:)=hldstr(j:k)
      hold(2)(1:)=instrg(j:k)
c
      if (hold(1).ne.hold(2)) go to 60
      ictr=ictr+1
60    continue
      if (ictr.eq.length) go to 200
100   continue
      go to 9999
c
c           match
c
c
200   match=1
c
c           return
c
c
9999  return
      end
```



```

c
c
c
c
c
c      k g s u p
c
c
c
c
c
c
c
c
c
c      subroutine kgsup (istat)
c
c      written by judith calabrese - 13 october 1981
c      fortran77/unix conversion by kathy miller - dec 83
c
c      provides the update mode of the kgs retrieval system.
c
c      subroutine called from kgsedit
c
c          ***** logical unit assignments *****
c
c      lu1 - kgs author index file (sequential)
c      lu2 - kgs author dictionary file (opened in kgsptr)
c      lu5 - standard input device
c      lu6 - standard output device
c      lu7 - kgs keyword index file (sequential)
c      lu8 - kgs keyword dictionary file (opened in kgsptr)
c      lu9 - kgs data base master file (direct)
c
c          kgs data base data elements
c
c      character author*32(3)
c      character pubdat*12
c      character title*100
c      character pubnam*32
c      character refer*20
c      character keywrd*16(6)
c      character discpn*16
c
c      character hauthr*32(3)
c      character hldkey*16(6)
c
c      character dauthr*32(3)
c      character diskey*16(6)
c
c      character buffer*4
c      character title1*78
c      character title2*35
c
c      character autidx(30)
c      character wrdidx(30)
c
c      character spaces*2
c      character astrix
c      character iresp*4

```

```

c      integer*2 autptr(30), wrdptr(30), authct, wrdct
c      integer*2 irec, iline, itype, endrec
c
c          initialize values
c
c      spaces=' '
c      astrix='*'
c      authct=0
c      wrdct=0
c
c
c          open kgs index files - author & keyword
c
c      open (1,iostat=istat,err=8010,file='/usr/kgs/data/authidx',
-      recl=5,form='formatted')
c      open (7,iostat=istat,err=8010,file='/usr/kgs/data/keyidx',
-      recl=5,form='formatted')
c
c          build index arrays
c          for dictionary search
c
c      do 2 i=1,30
c      read (1,9000,iostat=istat,end=3) autidx(i),autptr(i)
c
c          check for end-of-file condition
c      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 3
c
c      authct=authct+1
c      continue
c      do 5 i=1,30
c      read (7,9000,iostat=istat,end=10) wrdidx(i),wrdptr(i)
c
c          check for end-of-file condition
c      if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 10
c
c      wrdct=wrdct+1
c      continue
c
c          close indexes
c
c      close (1)
c      close (7)
c
c
c          open kgs data base
c
c      open (9,iostat=istat,err=8010,file='/usr/kgs/data/kgsmaster',
-      recl=372,access='direct',form='formatted')
c
c          find end of file
c
c      endrec=1
c      read(9,9005,iostat=istat,rec=endrec) author(1)
c
c          check for end-of-file condition

```

```

        if (istat.eq.-1.or.istat.eq.31.or.istat.eq.32) go to 17
c
        endrec=endrec+1
        go to 15
17      continue
        endrec=endrec-1
c          ***** update mode *****
c
c          user enters record # to be updated
c
20      write (6,9010,err=8000)
        read (5,9020,err=8000) iresp
        if (iresp.eq.astrix) go to 9999
        if (iresp.eq.spaces) go to 9999
c
c          convert record # to integer value
c
        call convrt (iresp,irec)
c
c          check for record past end-of-file
c
        if (irec.gt.endrec) go to 8060
c
c          read kgs data base record
c
        read (9,9030,iostat=istat,err=8020,rec=irec)
-      (author(i),i=1,3),pubdat,title,pubnam,
-      refer,(keywrd(j),j=1,6),discpn
c
        do 30 i=1,3
        if(author(i).eq.'deledeledeledeledeledeledeledele')go to 8070
        dauthr(i)=author(i)
30      continue
        do 35 i=1,6
        diskey(i)=keywrd(i)
35      continue
c
c          initialize author and keyword fields to spaces
c
        do 70 i=1,3
        hauthr(i)=spaces
70      continue
        do 80 i=1,6
        hldkey(i)=spaces
80      continue
c
c          display record
c
40      call kgsdln(dauthr,pubdat,title,pubnam,
-      refer,diskey,discpn,irec)
c
c          enter line # to be updated
c
50      write (6,9040,err=8000)

```

```

      read (5,9020,err=8000) iresp
      if (iresp.eq.astrix) go to 2000
      if (iresp.eq.spaces) go to 2000
      call convrt (iresp,iline)
c
c          check for valid line #
c
      if (iline.lt.1.or.iline.gt.14) go to 8030
c
c          update based on line selection
c
      go to (100,200,300,400,500,600,700,800,
-      800,800,800,800,800,1400),iline
c
c
c          update first author field
c
100  write (6,9050,err=8000) author(1)
      read (5,9060,err=8000) hauthr(1),buffer
      if (buffer.ne.spaces) go to 8040
      dauthr(1)=hauthr(1)
      go to 40
c
c          update second author field
c
200  write (6,9070,err=8000) author(2)
      read (5,9060,err=8000) hauthr(2),buffer
      if (buffer.ne.spaces) go to 8040
      dauthr(2)=hauthr(2)
      go to 40
c
c          update third author field
c
300  write (6,9080,err=8000) author(3)
      read (5,9060,err=8000) hauthr(3),buffer
      if (buffer.ne.spaces) go to 8040
      dauthr(3)=hauthr(3)
      go to 40
c
c          update publication date
c
400  write (6,9090,err=8000) pubdat
      read (5,9100,err=8000) pubdat,buffer
      if (buffer.ne.spaces) go to 8040
      go to 40
c
c          update title field
c
500  title1(1:65)=title(1:65)
      title2(1:35)=title(66:100)
c
520  write (6,9110,err=8000) title1,title2
      read (5,9120,err=8000) title1
      read (5,9130,err=8000) title2,buffer
      if (title2.eq.spaces) go to 550

```

```

c          remove extra spaces from title field
call cram (title1,title2)
c          create 100 character title field
550 title(1:78)=title1(1:)
    title(79:)=title2(1:)
    go to 40

c
c          update publication name
c
600 write (6,9140,err=8000) pubnam
    read (5,9060,err=8000) pubnam,buffer
    if (buffer.ne.spaces) go to 8040
    go to 40

c
c          update reference field
c
700 write (6,9150,err=8000) refer
    read (5,9160,err=8000) refer,buffer
    if (buffer.ne.spaces) go to 8040
    go to 40

c
c          update keyword fields
c
c
800 i=iline-7
820 write (6,9170,err=8000) keywrd(i)
    read (5,9180,err=8000) hldkey(i),buffer
    diskey(i)=hldkey(i)
    if (buffer.ne.spaces) go to 8040
    go to 40

c
c          update discipline field
c
1400 write (6,9190,err=8000) discpn
    read (5,9180,err=8000) discpn,buffer
    if (buffer.ne.spaces) go to 8040
    go to 40

c
c          save updated record?
c
c
2000 call kgsdln (dauthr,pubdat,title,pubnam,
-          refer,diskey,discpn,irec)
    write (6,9200,err=8000)
    read (5,9020,err=8000) iresp
    if (iresp.eq.'yes ') go to 2100
    if (iresp.eq.'no  ') go to 7000
    iline=1
    go to 8050

c
c          update dictionary files
c

```

```

c
c          setup arrays
c
2100 do 2140 i=1,3
      if(hauthr(i).eq.spaces.and.dauthr(i).ne.spaces)author(i)=spaces
2140 continue
c
      do 2160 i=1,6
      if(hldkey(i).eq.spaces.and.diskey(i).ne.spaces)keywrd(i)=spaces
2160 continue
c
c          delete old authors and keywords
c
c          itype=2
      call kgsptr (author,keywrd,irec,autidx,wrddix,
-              autptr,wrddptr.authct,wrddct,itype)
      if (itype.eq.9) go to 9999
c
c          add new authors to dictionary
c
c          itype=1
      call kgsptr (hauthr.hldkey.irec,autidx,wrddix,
-              autptr,wrddptr.authct,wrddct,itype)
      if (itype.eq.9) go to 9999
c
c
c          rewrite updated record
c
c
      write (9,9030,iostat=istat,err=8020,rec=irec)
-      (dauthr(i),i=1,3),pubdat.title,pubnam,
-      refer,(diskey(j),j=1,6),discpn
c
c
c          continue
c
c
7000 write (6,9210,err=8000)
      read (5,9020,err=8000) iresp
      if (iresp.eq.'yes ') go to 20
      iline=2
      if (iresp.ne.'no  ') go to 8050
      istat=0
c
c          closeout
c
9999 close (9)
      return
c
c
c          error reporting and formatting
c
c
8000 write (6,8005,err=9999)
8005 format (/1x,'ksup:command device error'/)
      istat=1

```

```

      go to 9999
8010 write (6,8015,err=8000) istat
8015 format (/1x,'ksup:error on opening file - iostat = ',i4/)
      go to 9999
8020 write (6,8025,err=8000) istat
8025 format (/1x,'ksup:error on reading file - iostat = ',i4/)
      go to 9999
8030 write (6,8035,err=8000)
8035 format (/1x,'line number is out of range - ',
-       1x,'please enter again'/)
      go to 50
8040 write (6,8045,err=8000)
8045 format (/1x,'line too long!!! - please enter again'/)
      go to (100,200,300,400,520,600,700,820,
-       820,820,820,820,820,1400),iline
8050 write (6,8055,err=8000)
8055 format (/1x,'please respond yes or no'/)
      go to (2000,7000),iline
8060 write (6,8065,err=8000) endrec
8065 format (/1x,'record entered is past end of file'/,
-       1x,'last record # is ',i4)
      go to 7000
8070 write (6,8075,err=8000)
8075 format (/1x,'record has been deleted'/)
      go to 7000

c
c
c       i/o formatting
c
c
9000 format (a1,i4)
9005 format (a32,a340)
9010 format (/1x,'enter # of record to be updated'/)
9020 format (a4)
9030 format (3a32,a12,a100,a32,a20,6a16,a16)
9040 format (/1x,'enter # of line to be updated',
-       1x,'or an * to end'/)
9050 format (/1x,'old first author is ',a32/,
-       1x,'enter new first author (doe. j) - lower case only',
-       1x,'in format: <last><,>< ><first>'//)
9060 format (a32,a4)
9070 format (/1x,'old second author is ',a32/,
-       1x,'enter new second author (doe. j) - lower case only',
-       1x,'in format: <last><,>< ><first>'//)
9080 format (/1x,'old third author is ',a32/,
-       1x,'enter new third author (doe. j) - lower case only',
-       1x,'in format: <last><,>< ><first>'//)
9090 format (/1x,'old publication date is ',a12/,
-       1x,'enter new publication date'//)
9100 format (a12,a4)
9110 format (/1x,'old title is ',/1x,a78/,1x,a35/,
-       1x,'enter new title (lower case only)'//)
9120 format (a78)
9130 format (a35,a4)
9140 format (/1x,'old publication name is ',a32/,
-       1x,'enter new publication name'//)

```

```
9150 format (/1x,'old reference locator is ',a20/,  
-      1x,'enter new reference locator in format:',//,1x,  
-      ' <ref owner><physical location><file number>'//)  
9160 format (a20,a4)  
9170 format (/1x,'old keyword is ',a16/,  
-      1x,'enter new keyword (lower case only)'//)  
9180 format (a16,a4)  
9190 format(/1x,'old discipline is ',a16/,1x,'enter new discipline'//)  
9200 format (/1x,'save o.k.?'//)  
9210 format (/1x,'do you want to continue in update mode?'//)  
end
```



```
c
c
c   program kgsupvr
c
c   written by judith calabrese - 19 october 1981
c   fortran77/unix conversion by kathy miller - dec 83
c
c   drives the kgs data base retrieval system
c
c
c   character iresp*4
c
c
c   lu5 - standard input device
c   lu6 - standard output device
c
c
c           header
c
c   write (6,9000,err=8000)
c
c           enter mode of operation
c
c   write (6,9010,err=8000)
50  read (5,9020,err=8000) iresp
c
c   if (iresp.eq.'edit') go to 100
c   if (iresp.eq.'extr') go to 200
c   if (iresp.eq.'info') go to 300
c   if (iresp.eq.'stop') go to 9999
c   if (iresp.eq.'    ') go to 9999
c   go to 8010
c
c           edit mode
c
c   100 call kgsed (istat)
c      if (istat.ne.0) go to 9999
c      go to 50
c
c           extract mode
c
c   200 call kgsext (istat)
c      if (istat.ne.0) go to 9999
c      go to 50
c
c           information mode
c
c   300 call kgsinf (istat)
c      if (istat.ne.0) go to 9999
c      go to 50
c
c           stop
c
```

```

9999 stop
c
c
c          error reporting and formatting
c
8000 write (6,8005,err=9999)
8005 format (/1x,'ksupvr:command device error')
      go to 9999
8010 write (6,8015,err=8000)
8015 format (/1x,'invalid response - please enter again'/)
      go to 50
c
c
c          i/o formatting
c
9000 format (//////////10x,'k g s   p u b l i c a t i o n s   ',
-          'd a t a   b a s e'//10x,
-          '#####'//)
9010 format (////////20x,'enter desired run mode'///,
-          10x,'edit      extract      information      stop'//)
9020 format (a4)
      end

```

```
# kgsrenew
#
# command file rewrites the kgs data base
#
# removes deleted records
#
# creates author and keyword dictionaries
#
# creates author and keyword indexes
#
#
# copy kgs data base master file into hold file
#
cp /usr/kgs/data/kgsmaster /usr/kgs/data/kgshold
rm /usr/kgs/data/kgstemp
#
# remove deleted records
#
cmd/krenew
#
#
mv /usr/kgs/data/kgstemp /usr/kgs/data/kgsmaster
#
#
# create dictionaries and indexes
#
rm /usr/kgs/data/kgsdictdump
rm /usr/kgs/data/authdict
rm /usr/kgs/data/authidx
rm /usr/kgs/data/keydict
rm /usr/kgs/data/keyidx
#
cmd/keny
#
# if program terminated normally
# please delete kgshold file
#
# type rm /usr/kgs/data/kgshold
#
#
```

```
#
#
#
# kgs
#
# loads the kgs program
#
#
# the extract mode includes
#   author, key word or all
#
# the edit mode includes
#   add, delete or update
#
rm /usr/kgs/data/printfile
#
cmd/kgs
#
# provide carriage control conversion
#
ex /usr/kgs/data/printfile << !
1,$s/~L/~L/
w
q
!
#
# print kgs printfile
pr /usr/kgs/data/printfile
#
```

Bibliography

Bourne, S.R. "An Introduction to the UNIX Shell." Bell Laboratories, Murray Hill, NJ.

Bourne, S.R. The UNIX System. Reading, MA: Addison-Wesley Publishing Company, 1982.

Calabrese, Judith T., Lawrence J. Kaetzel, Robert A. Glass & George R. Smith. A Computer Data Base System for Indexing Research Papers. NBS Technical Note 1167: U.S. Government Printing Office, 1982.

Chen, Ching-chih. Online Bibliographic Searching. New York: Neal-Schuman Publishers, Inc., 1981.

Chorafas, Dimitris N. Office Automation: The Productivity Challenge. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1980.

Date, C.J. An Introduction to Database Systems - Volume II. Reading, MA: Addison-Wesley Pub. Co., 1983.

Feldman, S.I. and P.J. Weinberger. "A Portable Fortran 77 Compiler." Bell Laboratories: Murray Hill, NJ, 1978.

Heaps, H.S. Information Retrieval: Computational and Theoretical Aspects. New York: Academic Press, 1978.

OS/32 Operator Reference Manual. Perkin-Elmer (Interdata Division), Oceanport, NJ, 1978.

Pressman, Roger S. Software Engineering: A Practitioner's Approach. New York: McGraw-Hill Book Company, 1982.

Pylyshyn, Zenon W. Perspectives on the Computer Revolution. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1970.

Salton, Gerard, and Michael J. McGill. Introduction to Modern Information Retrieval. New York: McGraw-Hill Book Company, 1983.

Ullman, Jeffrey D. Principles of Database Systems.
Rockville, MD: Computer Science Press, Inc., 1980.

AN INTERACTIVE BIBLIOGRAPHIC REFERENCE SYSTEM

by

KATHLEEN ANN MILLER

BA, University of Southern Colorado, 1973

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

-1984-

ABSTRACT

In our world of information explosion, professionals need tools with which to organize their research materials. In this report, several system models are examined for their suitability as such a tool. One specific system, the KGS data base reference system, is found that can provide such facilities for the users - KSU Computer Science Department faculty. KGS will allow individual faculty members to maintain a data base of on-site reference materials which may be retrieved by author(s), keyword(s) or title. Software file structures, operations and maintenance procedures are described.

This report also covers the implementation and enhancement of KGS. This implementation has involved conversion of the software from Fortran VII to Fortran 77. It has also necessitated changes from the operating system under which the original KGS system was run, OS/32 MT, to the KSU Computer Science's current UNIX operating system on the Perkin-Elmer 8/32.

The final form of KGS is described in detail, including a discussion of the implementation activities and results, directions for future work, a listing of the complete source code, and a users' guide.