IMPLEMENTATION OF AN OFFICE INFORMATION SYSTEM FOR THE DEPARTMENT OF COMPUTER SCIENCE

by

JOHN SCOTT ANDERSON

B.S., Kansas State University, 1979

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

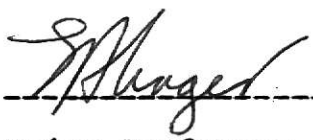
Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1983

Approved by:



Major Professor

TABLE OF CONTENTS

THIS BOOK CONTAINS NUMEROUS PAGES WITH THE ORIGINAL PRINTING BEING SKEWED DIFFERENTLY FROM THE TOP OF THE PAGE TO THE BOTTOM.

THIS IS AS RECEIVED FROM THE CUSTOMER.

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.

I would like to thank Mary Beth Cole for her input and patience in the Computer Science Department Office.

I would also like to thank Michael Terry for the design and organization of the "KOOL RECORDS SYSTEM".

I would also like to thank Carlos Qualls for his technical help and without whom this project couldn't have began, and thanks should also go to Harvard Townsend without whom this project couldn't have finished.

And most of all I would like to give special thanks to Dr. Elizabeth Unger, whom without her guidance and support the project could not have been started or completed.

CHAPTER ONE

INTRODUCTION

1.1 OBJECTIVE

The objective of this report is to describe the implementation of an office information system (OIS) for the Department of Computer Science. The intent of this system is to allow the department faculty and staff to monitor the progress of its applicants for graduate study and its graduate students. This office information system, termed the "KOOL RECORDS SYSTEM", consists of a menu-driven transaction processor interfacing with a database and report generator. The "KOOL RECORDS SYSTEM" is written in the C programming language and incorporates Unix operating system library software (2,4,5,7). The "KOOL RECORDS SYSTEM" will reside on the department's Perkin-Elmer 3220 minicomputer system and operate under the Unix operating system.

1.2 BACKGROUND

In the spring of 1983, Michael S. Terry completed a master's report detailing the design of an office information system consisting of a transaction processor coupled with a relational database and report generator (6). He used a functional approach to incorporate the already

existent routines of the Computer Science Department faculty and staff into the system that was intended to track the progress of graduate students and applicants.

An emphasis was placed on the system design to be as "user-friendly" as possible. This meant that the system must interface with the user in such a way as to be easily understandable, easy to use, and expeditious. For this to occur, Mr. Terry conducted extensive interviews with the appropriate personnel (the future system users) to determine the functions for the system to perform, and to develop an understanding of these functions. Once this was achieved, he then designed the system screens that were to be used for input and perusal of student information. (A screen can be defined simply as a form or format that appears to the user on a CRT terminal). These screens were to be used in a menu-driven fashion to interface with the relational database and with the report generator.

The relational database, as designed, consisted of two separate relational databases. One of these databases was to contain the information for each of the Computer Science Department's graduate students, while the other database contained the information for each of the department's graduate study applicants. These databases were designed to consist of normalized relations. Mr. Terry verified, with the use of Bernstein's Algorithm II, that these relations were in the third normal form. He also verified that the

database relations were in the Boyce-Codd normal form, as well as fourth and fifth normal forms.

It has been the intent of the author to implement the "KOOL RECORDS SYSTEM". Since Terry's methodology included working closely with the future system users, it is felt that his design adequately met with their requirements. Because of this, every attempt has been made to continue the communication with future users and to adhere strictly to the design during the implementation phase of the "KOOL RECORDS SYSTEM".

1.3 REPORT OUTLINE

This report will detail the entire implementation phase of the "KOOL RECORDS SYSTEM".

Chapter Two provides the components of the "KOOL RECORDS SYSTEM" and describes how these components are structured. This chapter includes a discussion of the system design and implementation. The data structures that are used are also discussed.

Chapter Three describes the "KOOL RECORDS SYSTEM" in a diagrammatic manner. Included is a brief introduction to a new software diagram methodology that is used to detail the interactions that occur between the user, the data, and the programs.

Chapter Four provides a critique of the implemented "KOOL RECORDS SYSTEM"
and suggests possible enhancements that could be added in the future.

In addition to the information contained in the above chapters, Appendix D contains a users manual that outlines the steps required of a user to perform the various functions that the "KOOL RECORDS SYSTEM" provides.

CHAPTER TWO

PROGRAM AND DATA STRUCTURES

2.1 OVERVIEW

The intent of this chapter is to describe, in detail, the architecture of the "KOOL RECORDS SYSTEM" along with the various data structures supported by this system. This description will include details of the implementation of this system. Numerous references will be made to the design phase of the "KOOL RECORDS SYSTEM" project. To aid the transition of the design phase to the implementation phase and to better illustrate the architecture and structures involved in both phases, Warnier/Orr diagrams will be employed throughout this chapter. The following section will briefly describe Warnier/Orr diagrams.

2.2 DESCRIPTION OF WARNIER/ORR DIAGRAMS

The Warnier diagram was first developed by Jean-Dominique Warnier and was discussed in "The Logical Construction of Programs" authored by Warnier (8). Kenneth Orr further modified the diagram to encompass both data flow and data structure-oriented design methodologies.

The Warnier/Orr diagram describes the logical organization of information in a hierarchical system.

Instead of the common vertical hierarchical block diagram, the Warnier/Orr diagram flows horizontally from left to right and involves the use of braces instead of boxes and arcs or lines. The diagram can be thought of as a mathematical set being divided into subsets (with the use of the brace) and each subset being further divided until the smallest minimal subset is reached.

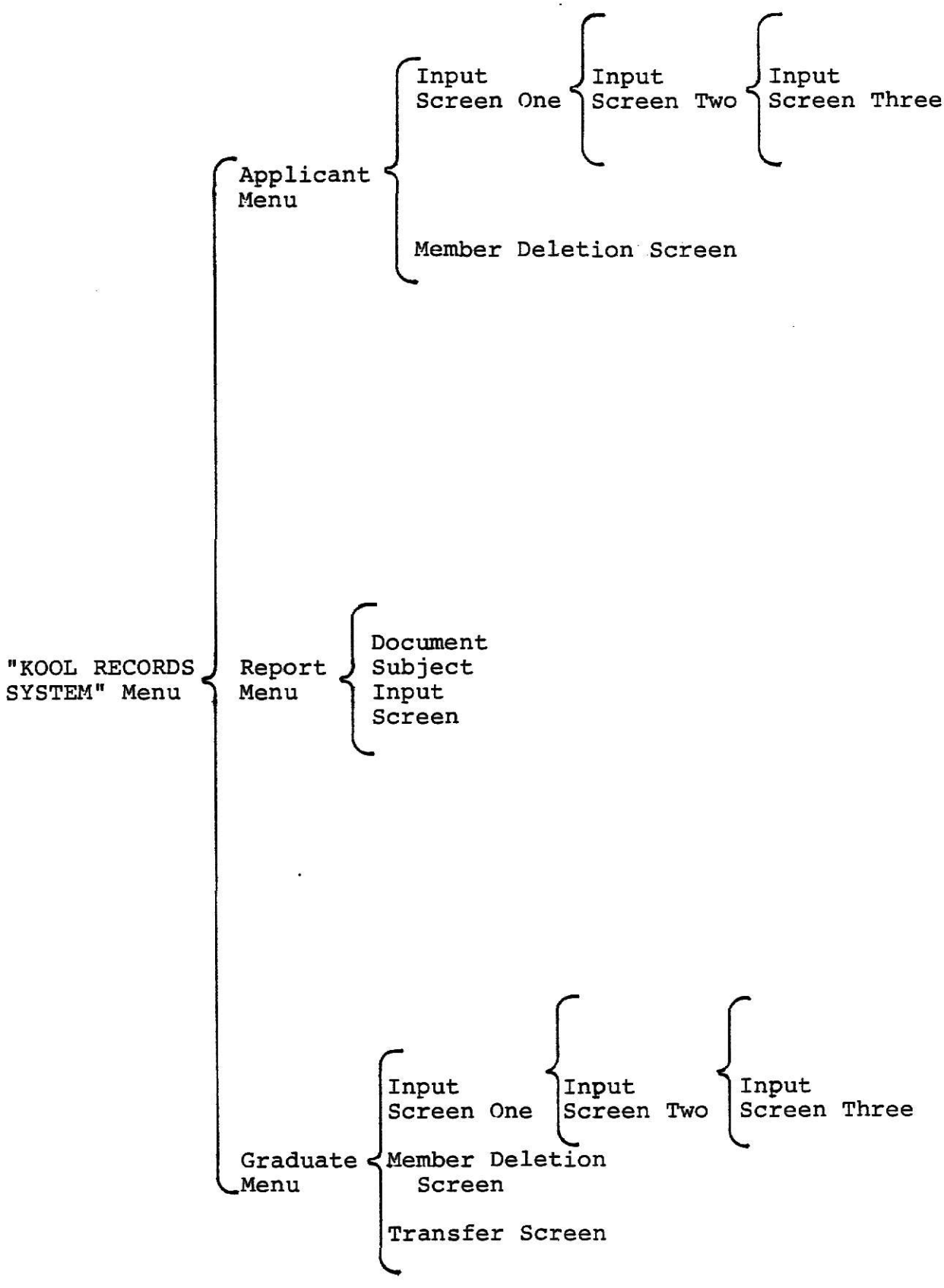
Warnier/Orr diagrams will be used to provide the overall structure of the "KOOL RECORDS SYSTEM" components.

2.3 TRANSACTION PROCESSOR STRUCTURE

The transaction processor, as designed by Michael Terry, includes a series of screens that are to allow information insertion to, deletion from, and perusal of the applicant and graduate databases. In addition, screen formats exist to allow the transfer of pertinent information from the applicant database to the graduate database as applicants are accepted into the graduate study program. Screens to allow the staff to print documents and reports as a function of the system's report generator are an additional part of the designed system.

The transaction processor's flow of control through these screens is illustrated in the Warnier/Orr diagram in Figure 2.1. This diagram depicts, in a hierarchical fashion, the order of these screens as the user navigates from a "main-

Figure 2.1


"KOOL RECORDS SYSTEM" Menu
Applicant Menu
Input Screen One
Input Screen Two
Input Screen Three
Member Deletion Screen
Report Menu
Document Subject Input Screen
Graduate Menu
Input Screen One
Input Screen Two
Input Screen Three
Member Deletion Screen
Transfer Screen

menu" to the "sub-menus" and finally to the input screens that act as the interface between the user and the applicant and graduate databases. Each of these input screens also allows information deletion, as well as editing and perusal. The member deletion screens were designed to allow complete removal of a student or applicant's record from the respective database. The transfer screen allows information to be copied from the applicant database to the graduate database. This diagram of the system screens also conveniently serves as a template for the structure of the various programs of the "KOOL RECORDS SYSTEM". This template served as an aid to the implementation phase of the project by demonstrating the modularization of the system's functions. The system's functional modules will be discussed further in later paragraphs.

The actual implementation of the transaction processor of the "KOOL RECORD SYSTEM" can be divided into eleven steps. These steps are as follows:

(1) generation of system screens

(2) development of routines to allow navigation through system screens

(3) development of routines to allow user input of information

(4) creation of temporary data structures to contain information entered by the user

(5) development of code to allow transfer of information from temporary data structures to databases

(6) development of code to allow retrieval of information from databases

(7) development of code to allow transfer of information from one database to the other

(8) development of record deletion function

(9) development of code for user verification for system security.

(10) development of faculty perusal function

(11) development of rejected applicant function

Each of these steps will be described in detail in the following sections including illustrations to demonstrate the structure of the transaction processor.

2.3.1 GENERATION OF "KOOL RECORDS SYSTEM" SCREENS

The first step of the implementation phase of the project was to devise a method to generate the designed screens on the CRT terminal. Various methods were attempted including placing the screen on the CRT, first in a "character by character" fashion, and then attempting a "string by string" approach. These methods proved to be too long and tedious, in the sense of the amount of code that was produced, and thus it was felt, these methods were unconstructive. Finally, a more expedient approach was discovered. Since these screens were already "written" on a Unix file during the design phase, they were already within the required eighty column, twenty-five line CRT terminal screen size. The screen generation technique simply involved reading and

writing the Unix file onto the CRT terminal screen. This procedure was delightfully effortless since it only required approximately five lines of code to generate each of the seventeen screens of the "KOOL RECORDS SYSTEM". The source code for this screen generation method can be found in Appendix A-1 under the program name "screen.c".

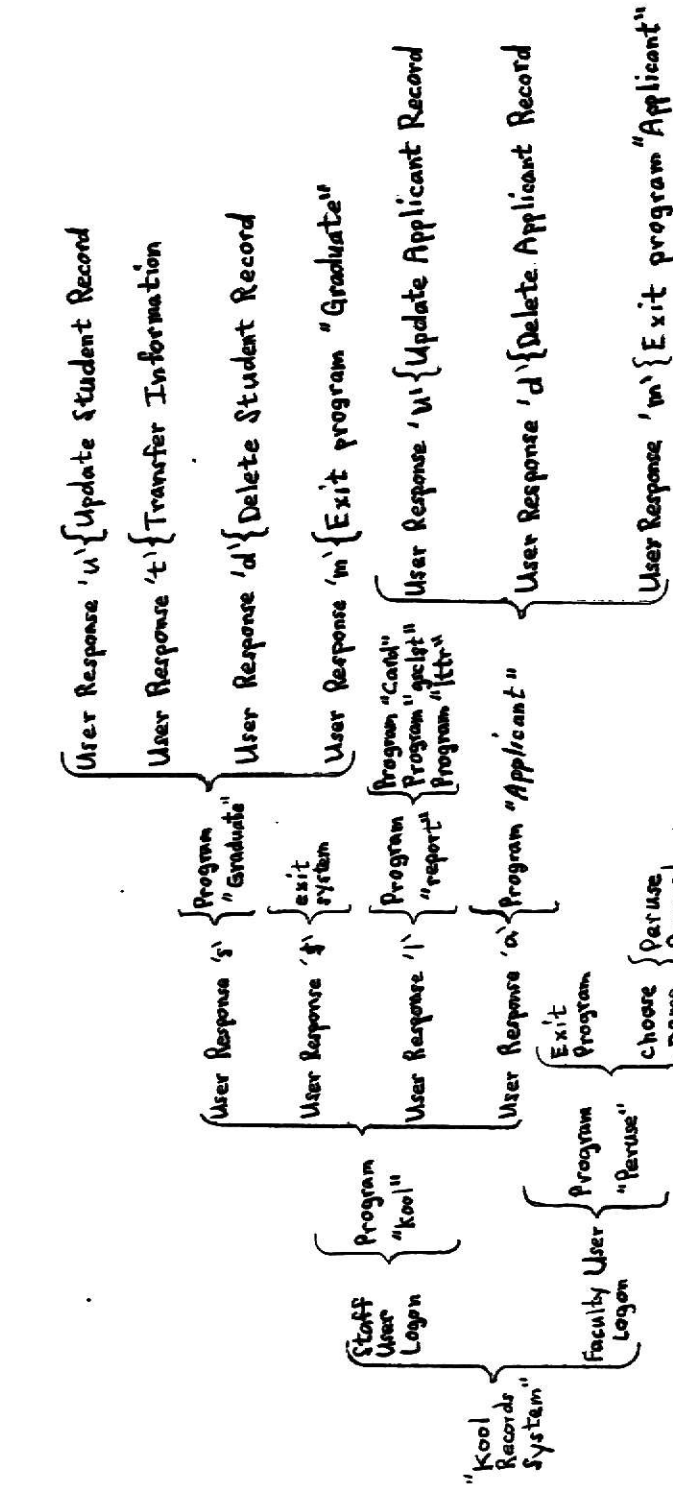
2.3.2 DEVELOPMENT OF SCREEN NAVIGATION ROUTINES

Once the code for the generation of system screens was implemented, routines were needed to allow the user to navigate from the "main menu" to each of the input screens and back again to the "main menu". Since this system was to be as "user-friendly" as possible, a minimum of user keystrokes was desired. The system screens, as designed by Mr. Terry, required only a single keystroke. Also, to promote user-friendliness, the appropriate key for the user to press was mnemonically designed. For example, a user response of 'm' will bring the user back to the menu or sub-menu of the current program (either graduate program, applicant program, transfer program, or deletion program). During the implementation phase, user-friendliness was also promoted by accepting either capitalized or uncapitalized letter commands as valid input.

Navigation through the system was actually implemented by allowing the user to initiate the desired program by simply pressing the appropriate key. If an incorrect key is pressed

by the user, a friendly error message appears stating that the user must make another attempt. The number of incorrect entries allowed is left to the user's discretion. The navigational flow through the system is illustrated in Figure 2.2. Tracing this flow, a log on to the system by the appropriate user initiates program "kool". From this point a user response of 's', for student, will initiate program "graduate", user response 'a' will initiate program "applicant", and user response 'l', for letter, will initiate program "report". Figure 2.2 then, depicts the required response of the user to navigate further into the system. Figure 2.2 can be compared to Figure 2.1 to correlate the screens associated with the various programs.

These user responses, as described in the above paragraph, are actually programmed as components of a control flow statement in the C language. This statement is called a switch statement in C and is analogous to the case statement in Pascal. Each user response is actually a "case" within this switch statement. Each case, then, either initiates a procedure within the current program, or allows a system call to allow execution of another separate program, as can be seen in Figure 2.2. These switch statements can be found throughout the "KOOL RECORDS SYSTEM" source code, conveniently and logically specifying the flow of control as dictated by the user.


"Kool Records System"
Staff User Logon
Program "kool"
Faculty User Logon
Program "Peruse"
User Response 's' Program "Graduate"
User Response 'x' exit system
User Response 'r' Program "report"
User Response 'a' Program "Applicant"
Exit Program
choose name
Peruse Record
User Response 'u' {Update Student Record
User Response 't' {Transfer Information
User Response 'd' {Delete Student Record
User Response 'm' {Exit program "Graduate"
Program "card"
Program "script"
Program "ttt"
User Response 'u' {Update Applicant Record
User Response 'd' {Delete Applicant Record
User Response 'm' {Exit program "Applicant"

2.3.3 DEVELOPMENT OF USER INPUT ROUTINES

The system allows a user to "flash" screens onto a CRT in a menu-driven fashion. The user can very simply follow the directions given on each screen to traverse to the next screen or to "log off" of the system. User input routines allow the user to input information on a student or applicant. The manner in which the screens are designed dictate that the user input of information is analogous to "filling in a paper form". Each screen contains labelled fields corresponding to the labelled fields on a paper form. When filling in a paper form, the user moves the pen or pencil to the first field and fills in the appropriate information. The user then moves the pen or pencil to the next field and fills in that field, etc. With the use of a C library screen package, the cursor can be directed to any point on the screen (1). This is accomplished using the following statement provided with the screen package:

```
mvcur(old_y,old_x,new_y,new_x);
```

where "old_y" and "old_x" are the coordinates of the old cursor position on the CRT screen ("where you are"), and "new_y" and "new_x" are the coordinates of the new cursor postion on the screen ("where you want to go"). In order for a C program to have access to this screen package the preprocessor command line: " include <curses.h>" must be

added to the program's declaration section. In the "KOOL RECORDS SYSTEM", cursor movement is triggered when the user presses the 'return' key. Once the user has directed the cursor to the desired field in a "top-to-bottom" fashion, information can be entered at that location. Simple editing procedures are performed by allowing the user to backspace to the mistake, correct it, and then retype the rest of the information. This can only be done, however, if the cursor is already located in the field containing the mistake.

2.3.4 TEMPORARY DATA STRUCTURES

Two main structures are used in the "KOOL RECORDS SYSTEM". One of these, found in Appendix A-12under the file name "appstrt.h", is termed "temp_rec". The other, "grad_rec", is found in Appendix A-13 under the file name "grstrt.h". These structures (a C language term) are analogous to records in Pascal. When the return key is pressed by the user to go from one field to another, the information just entered is copied into one of these temporary data structures, depending on which program is executing. These structures consist of fields (analogous to a record field in Pascal) that correspond to the field labels on the screen. For example, the screen field label "NAME" corresponds to the C language structure variable "temprec.name" in the applicant program, "app.c" found in Appendix A-3, or "gradrec.name" in the graduate program

"grad.c" found in Appendix A-4. Each field on the screen then has its corresponding structure field for information to be copied to as it is entered. These structure field names also correspond to the attribute names of the database relations in the "KOOL RECORDS SYSTEM" design.

2.3.5 INSERTION OF INFORMATION INTO THE DATABASES

Each time the user enters information into the temporary data structures through use of the input screens, this information must be filed into the appropriate database. This filing of information occurs as the user returns to the program's menu (either the applicant program or the graduate program). The filing procedure is accomplished with the use of C's "write" statement along with two other key statements, "open" and "lseek". The write statement, which is actually a system call, is a low level I/O Unix function. This call provides direct entry into the operating system. The write call has the following format:

```
n_written = write(fd,buf,n);
```

where "fd" is the file descriptor, "buf" is the data structure (either temprec or gradrec) and "n" is the number of bytes to be transferred. Each call returns the number of bytes transferred, which is "n_written" in the

above statement. The actual call format used in the "KOOL RECORDS SYSTEM" program "app.c" for example, is:

```
n_written = write(fd,&temprec,sizeof(struct temp_rec));
```

"&temprec" is the beginning address of the instance of the record structure temp_rec. The transfer of data will begin at this address and continue until the end of the structure is reached. The length if this structure is determined by the "sizeof()" function, which simply returns the size of whatever is inside the parentheses.

The "open" statement is also a system call found in C. This call opens the Unix file for either read access, write access, or read and write access. The type of access that is required at this point is write access. The format of the system call would then be as follows:

```
fd = open("database", 1);
```

where "fd" is the file descriptor, ""database"" is the name of the database, and "1" which only allows information to be written to the "database". If read access was required, a "0" would be used in place of "1". If both read and write access were required, the integer "2" would be used. Also, the file descriptor "fd" is the same "fd" found in the

"write" system call previously described.

The third major C statement used in the information transfer procedure is the "lseek" system call. "lseek" allows navigation within a Unix file without actually reading or writing. "lseek" can be used to pinpoint the location in the file of where reading or writing is to begin. The format of "lseek" is as follows:

```
lseek(fd, offset, origin);
```

The file descriptor "fd" is the same "fd" as returned by the "open" system call. The "lseek" call causes the current location within a file specified by "fd" to move to the location specified by "offset" beginning at a certain point "origin". This call is used in the "KOOL RECORDS SYSTEM" to find the specific point in a file to begin writing or reading information.

The following code segment, then, demonstrates the typical write procedure in the "KOOL RECORDS SYSTEM":

```
fd = open("database",1);
/* open database file for write only access */
lseek(fd,offset,0);
/* go to location offset starting at the beginning of the
file */
```

```
n_written = write(fd,&temprec,sizeof(struct temp_rec));
/* at location offset, write the temprec structure */
```

After the above routine is completed, the opened Unix file should be closed. This is accomplished using the following system call:

```
close(fd);
```

2.3.6 INFORMATION RETRIEVAL

Information retrieval from the databases is accomplished in much the same way as information "filing". The same system calls as described in section 2.3.5 are used except for the "write" call. This is replaced by the "read" system call. The typical program code segment to read information from the desired database is as follows:

```
fd = open("database",0);
lseek(fd, offset, 0);
n_read = read(fd,&temprec,sizeof(struct temp_rec));
```

The only difference here is that a zero is used in the open statement to provide read access only. And, of course, "read" and "n_read" replace "write" and "n_written",

respectively. Execution of this code, then, copies information from the Unix database file into the temporary record structure for subsequent display on the "KOOL RECORDS SYSTEM" screens.

2.3.7 DATABASE TO DATABASE INFORMATION TRANSFER

As an applicant is accepted into the graduate program, pertinent information from the applicant's files are needed. This information must be filed into the graduate files as this applicant becomes a graduate student. Because of this, a method was needed to transfer information from the applicant database to the graduate database. The following paragraphs describe this procedure.

The same Unix system calls described in section 2.3.6 are used for the information transfer procedure. To initiate these calls also requires user response. The basic steps that occur to transfer information from one database to the other are as follows:

(1) user input of names of new graduate students

(2) locate and read information on each entered name from the applicant database into a temporary applicant record structure.

(3) actual assignment of desired applicant information to a temporary graduate record structure

(4) locate correct position in graduate database and file the new graduate student's information.