The Design and Implementation of an Online Directory Assistance System

Kimberle Koile

December 1983

© Massachusetts Institute of Technology 1983

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Office of Naval Research under contract number N00014-75-C-0661.

Massachusetts Institute of Technology
Laboratory for Computer Science
Cambridge, Massachusetts 02139

The Design and Implementation of an Online Directory Assistance System

by

Kimberle Koile

Submitted to the
Department of Electrical Engineering and Computer Science
on December 20, 1983 in partial fulfillment of the requirements
for the Degree of Master of Science

Abstract

This thesis describes the design and implementation of an online directory assistance system called DIRSYS that was modeled after the white pages of a paper telephone book and a full-screen display editor such as Emacs. As the user begins typing a name, the "pages" of this electronic telephone book appear on the screen, and the entry that most closely matches what the user has typed so far is highlighted. The system provides a tutorial for novice users and an online help facility for novices as well as experienced users. The system also provides a facility for keeping the information in the database up-to-date. A preliminary evaluation of DIRSYS indicates that the system can be used easily by both inexperienced and experienced computer users and that, except for its slow performance, DIRSYS is usable and robust.

Keywords: directory assistance, name server, user interface.

Acknowledgments

I would like to thank my thesis advisor, Prof. Jerry Saltzer, for his advice and encouragement throughout this research and his diligence in reading the drafts of my thesis. I also would like to thank Prof. Tom Malone for reading a draft of my thesis and making valuable comments about it.

I would like to thank Sam Hsu for his invaluable help in designing and implementing this system and in clarifying the ideas in this thesis. I also would like to thank Deborah Estrin and Karen Sollins for their helpful suggestions. Thanks go to Sam, Deborah, and Karen for helping make my stay at M.I.T. an enjoyable one.

In addition, I would like to thank the other members of the Computer Systems and Communications Group and the Computer Systems Research Group for their comments and suggestions about my work. In particular, Larry Allen and Michael Greenwald have helped me learn about UNIX and VAXes. Also, thanks go to David Feldmeier for help with the Altos.

I also would like to thank those people, too numerous to list, who provided me with the needed directory information, as well as those who helped critique the system.

Special thanks go to my parents, Carmon and Earl, for their continued confidence and support. Thanks also go to my father for his valuable editing comments.

Finally, special thanks go to my husband, John, for his never-ending patience and support. I also am grateful to him for making many helpful editing suggestions and doing the graphs for this thesis.

Table of Contents

Chapter One: Introduction 6

1.1 System Goals 7
1.2 Related Work 8
1.3 Thesis Organization 18

Chapter Two: Operational and Design Issues 19

2.1 Operational Issues 19
2.2 Design Issues 23

Chapter Three: The System 27

3.1 The Interface 27
3.1.1 Design Principles 27
3.1.2 What the Interface Looks Like 31
3.1.3 Features that Illustrate Design Principles 49
3.2 The Database 52
3.2.1 Database and Index Structures 52
3.2.2 Searching the Database 53
3.3 The Update File 56
3.3.1 Update File Structure 56
3.3.2 Adding Updates to the Database 57

Chapter Four: Preliminary Evaluation 63

4.1 User Interface 63
4.2 Database Access Method 70

Chapter Five: Conclusions and Future Work 74

5.1 Conclusions 74
5.2 Future Work 75
5.2.1 Work on the Current System 75
5.2.2 Extensions to the Current System 78

Appendix A: Glossary of Terms 86

Appendix B: Performance Graphs 90

Bibliography 101

Table of Figures

Figure 1-1: Key Arrangement on a Dvorak Keyboard 9
Figure 3-1: Herald Screen 32
Figure 3-2: Sample Search Screen 33
Figure 3-3: Sample Search Screen in Expanded Format 36
Figure 3-4: Sample Tutorial Screen 38
Figure 3-5: Sample First Level Help Window 39
Figure 3-6: Sample Second Level Help Window 40
Figure 3-7: Sample Update Request Window 42
Figure 3-8: Sample Update Request Window Showing Proposed Changes 43
Figure 3-9: Sample Update Survey Window 45
Figure 3-10: Sample Update Survey Window in Expanded Format 47
Figure 3-11: Sample Update Edit Window 48
Figure 3-12: Sample Database and Indexes 55
Figure 3-13: Sample Update Record 57
Figure 3-14: Sample Update File and Database Before Daemon Runs 61
Figure 3-15: New Update File and Database After Daemon Runs 62
Figure 4-1: Disk Accesses vs. Record Block Size, Buffer Size 128 72
Figure 4-2: Central Processor Time vs. Record Block Size, Buffer Size 128 73
Figure 5-1: Sample Search Screen for Mouse Interface 82
Figure B-1: Disk Accesses vs. Record Block Size, Buffer Size 128 91
Figure B-2: Disk Accesses vs. Record Block Size, Buffer Size 256 92
Figure B-3: Disk Accesses vs. Record Block Size, Buffer Size 512 93
Figure B-4: Disk Accesses vs. Record Block Size, Buffer Size 1024 94
Figure B-5: Disk Accesses vs. Record Block Size, Buffer Size 2048 95
Figure B-6: Central Processor Time vs. Record Block Size, Buffer Size 128 96
Figure B-7: Central Processor Time vs. Record Block Size, Buffer Size 256 97
Figure B-8: Central Processor Time vs. Record Block Size, Buffer Size 512 98
Figure B-9: Central Processor Time vs. Record Block Size, Buffer Size 1024 99
Figure B-10: Central Processor Time vs. Record Block Size, Buffer Size 2048 100

Chapter One

Introduction

DIRSYS[1] is an online directory assistance system. It was developed for users with widely varying computer skills and is based upon the familiar concepts of a paper phone book and a full-screen display editor such as Emacs [51]. Entries from the directory are displayed on the screen in a compact format, one line per entry, and the current entry of interest is *highlighted*, e.g., by displaying the line in reverse video. The user may direct the system to emphasize another entry by issuing commands, similar to Emacs' cursor motion commands, or by typing a name. The search mechanism is *incremental*. That is, after each character typed by the user, DIRSYS updates its highlight and the terminal screen, if necessary, such that the highlight rests on the entry whose name string most closely matches what the user has typed so far. A *help facility* is provided to guide the novice user and to remind the experienced user which commands are available. A *tutorial* is also available for users who want step-by-step instruction on how to use DIRSYS. The default search screen allows approximately a full screen's worth of entries to be displayed, each entry occupying one line of the terminal screen. All information concerning a particular entry cannot be seen using this compact format. The user may request DIRSYS to display fewer entries on the screen and show each entry in detail. A command is available to switch between these two display formats. All commands retain their semantics regardless of the display format.

DIRSYS also provides a facility for keeping the information in the directory database up-to-date. A user may submit *update requests*, which contain proposed modifications to that user's database entry. The manager validates or invalidates

---

[1] The name is derived from DIRectory SYStem.

these requests, e.g., via letters to the users, and marks the valid update requests. DIRSYS then rebuilds the database to include the valid updates.

1.1 System Goals

DIRSYS was designed and implemented in order to make the task of looking up M.I.T. phone book information easier than with the current methods.[2] The primary goal for DIRSYS was to develop a system that could serve multiple classes of users -- inexperienced, experienced, and users in between who have mastered the system once but are not experienced at using it. Thus, the system needed to be easy to learn so that people unfamiliar with use of computers or use of the system could engage the system and "look someone up" in the electronic phone book. The system also needed to be easy to use once learned so that experienced computer users and experienced DIRSYS users could look up information easily and quickly. The challenge in designing DIRSYS was to combine the simplicity and verbosity needed for ease of learning with the flexibility and conciseness needed for ease of use.

In addition, it was desired (1) that the retrieval of information from the database be fast -- at least fast enough to keep up with the demands of the users, (2) that the database *access method*, i.e., the database structure and database search mechanism, be simple, and (3) that the *access time* for each database search, i.e., the time to search the database and retrieve the desired record, fall within a narrow range of values in order for the system response times to be relatively constant.

Another goal was to have a relatively simple update mechanism that would allow changes to be made to the directory database easily and quickly without degrading system performance.

---

[2]Currently, the telecommunications operators use microfiche and other individuals use paper phone books or online directory systems that contain information for users of certain computer systems.

1.2 Related Work

The following survey of online directory assistance systems is not an exhaustive survey, but rather a representative one. The systems outlined here range from small systems to very large systems, and most of them function only as directory assistance systems. One of the systems described here provides other services as well. Systems such as Xerox's Clearinghouse [40] and Grapevine [5] are designed to aid computers in locating objects in a distributed multinetwork environment. Since they do not provide the kind of directory assistance service for individuals that DIRSYS provides, they are not included in this survey.

French PTT Directory System

For the past several years, the French phone company (PTT) has been developing an online directory assistance system that is available to subscribers via terminals placed in their homes and offices and in post offices and selected public places [20]. This electronic telephone directory contains information found in the "white" and "yellow" pages of a printed telephone book. Users can search the directory for information by name, profession, or phone number, and can look up billing and telephone company information.

To search for information associated with a personal name or business name, the user types in a location name, where location may be a township, an urban precinct or district, a county, or a rural crossroads site, and the personal or business name. He[3] may also enter an address or the name of a profession, e.g., banking. The system uses this additional information to identify the desired entry when more than one matching entry is found. To search the "yellow" pages, the user inputs a location name and a qualifier that describes the information he is seeking, e.g., restaurant. If a matching entry is not found for the specified location, the program

---

[3]Unfortunately, English does not have a personal pronoun that is accepted to mean both he and she. Since forms such as "s/he" are not considered to be grammatically correct, "he" is used in this document to mean "he or she".

will search the information associated with neighboring areas. Finally, to search for an entry associated with a given telephone number, the user types in the eight digit national telephone number.

Directory assistance operators perform two functions within this system. They assist customers who are unable to locate a desired entry using the electronic directory system, and they update information contained in the directory database.

The online directory system is currently operating throughout the western part of France, with terminals available to any subscriber who wants one. In addition, plans are underway to introduce the system throughout the rest of the country.

New England Bell Directory System

The New England Bell Telephone Company's online directory assistance system was designed to be easily and quickly used by trained operators.[4] It is used to service requests for information contained in the "white pages" of phone books,[5] and employs a special keyboard, called a Dvorak keyboard, on which keys are located according to frequency of use, with the most frequently used keys in the middle row. This keyboard is used for quick input of query information. (See Figure 1-1.)

P Y F G C R L

A E I O U D H T N S

Q J K X B M W V Z

Figure 1-1: Key Arrangement on a Dvorak Keyboard

[4]Consequently, the system has no online help facility.

[5]There currently is no interactive system for looking up "yellow pages" requests.

To search for a phone book entry, an operator types in a two or three letter locality abbreviation, e.g., ARL for Arlington, a partition abbreviation, e.g., W for West Suburban Boston, three or four letters of a last name, one letter of a first name, and then hits a residential, business, or government key to indicate which type of phone book entry is being sought. The operators move the cursor between different input fields by using a <TAB> key. The entries that contain the typed characters in the appropriate fields are displayed in the format that is used in the paper phone books: one line per entry, last name followed by first and middle names or initials, street address, city abbreviation, and phone number. The system also provides facilities for locating entries containing similar, but differently spelled last names and names that phonetically match the query name.[6] The area code operators, reached by dialing 617-555-1212, typically service approximately 200,000 calls a day; the Boston area operators, reached by dialing 411, typically service approximately 260,000 calls a day.

The database for the 617 area code contains approximately 2.5 million entries and is partitioned into four sections; the database for the Boston area contains approximately 1.5 million entries and is partitioned into 16 sections.[7] Updates to the directory database are initiated by the customer, who calls a service representative to request a change. Once received, the updates are entered into an update file that is separate from the database. The database is rebuilt every night to include new update information.[8] Thus, updates are visible as soon as they are received without changing the database that is in use.

---

[6]These additional search facilities are used rarely by the directory assistance operators.

[7]Pat Martin, New England Bell Telephone Company, personal communication.

[8]The vendor in charge of maintaining the system delivers a magnetic tape containing the new database each morning.

CSNET Name Server

CSNET[9] is a computer communications network linking groups in the United States doing computer science research, such as university computer science departments. It provides, or will provide, electronic mail, file transfer, and remote login services to computers that are directly connected to CSNET and electronic mail services for computers not directly connected to CSNET. The CSNET Name Server, which provides a directory assistance service for CSNET users, was designed to be easy to use and to facilitate the sending of electronic mail by helping users locate addresses of mail recipients. In later stages, it will also help the user establish nicknames and aliases for mail recipients and forward electronic mail [27, 48].

The Name Server database currently contains about 2000 entries and is maintained, along with the programs for accessing it, on a central computer.[10] In addition, a Name Server program for registering users and answering questions about existing CSNET sites resides on computers at local sites. If the local computer is directly connected to CSNET, users invoke the directory service by connecting to the central computer and typing "ns". There are two levels of help available. The first level contains a list of commands and their functions, and the second level contains a more detailed explanation of a specified command.

To search the directory, a user types "whois" followed by a string of characters and optional keywords. For example, a user might type "whois James [Texas professor]", and the program would search the database for a record containing the string "James" and either of the keywords "Texas" or "professor". The program searches all fields in the directory records for a matching string. Records would be

---

[9] The name is derived from Computer Science NETwork.

[10] The computer has been moved recently from the University of Wisconsin at Madison to Bolt Beranek and Newman Inc., located in Cambridge, MA.

located that contain "James" as a first name, last name, login name[11], or street name, for example. The keywords are important for distinguishing between the records containing the same string. Typing an "*" in a string tells the program to match any character or characters in that position in the string. In this way the program can locate entries in the database when users do not know exactly how to spell the name for which they are searching. If there are fewer than nine entries matching what the user has typed, all information in them matching entries is displayed. Otherwise, only names and electronic mail addresses are displayed. There is also an option that allows users to specify this "short" format in place of the longer one.

If the local computer is not directly connected to CSNET, users may interact with the local Name Server program to formulate the "whois" queries. The program then sends the queries via electronic mail to the central Name Server computer, which performs the search and returns the matching information to the user via electronic mail.

Users are responsible for entering and updating their entries in the central directory database. The local Name Server program allows users to REGISTER, RETRIEVE, EDIT, INSTALL and UNREGISTER their directory information. The program formats an update message and sends it to the central Name Server computer. The new information is added to the directory database by an update program that runs every 15 minutes. Users may provide the Name Server program a password to be used when modifying or deleting their directory entry. Having a password for the entry allows a user to edit that entry when connected to the central Name Server, thus saving him from having to always interact with the local Name Server program.

---

[11]The term "login name" is often synonymous with "login ID", "computer account", or "electronic mail address".

NICNAME

NICNAME is a directory service for users of ARPANET[12], a computer communications network that links institutions throughout the United States, Norway, and England. NICNAME is maintained by the Network Information Center (NIC) and is invoked by connecting to the NIC computer and typing "whois" followed a string of characters [21]. If the string only contains an "*", an explanation about how to search for a name is displayed. Any other string of characters is interpreted as either a personal name or a handle, the unique identifier assigned to each entry in the directory database.[13] Typing "..." after a string of characters tells the program to find entries containing names or handles that begin with the typed characters, i.e., tells the program to use a prefix match rather than an exact match. Typing "." in front of a string of characters tells the program to search for matching names only; typing "!" in front of a string of characters tells the program to search for matching handles only; typing nothing before the string of characters tells the program to search for matching names and matching handles.

If only one matching entry is found, all of the information contained in the entry is displayed and usually occupies several lines on the screen. If several matching entries are found, only the names, handles, electronic mail addresses, and phone numbers in the entries are displayed, with the information for each entry occupying one line on the screen. If more than five matching entries are found, the first five are displayed on the screen, followed by the message, "There are *x* more entries. Show them? [Confirm]", where *x* is an integer. Hitting a carriage return or enter key causes the remaining matching entries to be displayed. To view more information for an entry that was displayed in the one line per entry format, the user repeats the search, typing enough of the name or handle to uniquely identify the desired entry.

---

[12]The name is derived from Advanced Research Projects Agency NETwork.

[13]The handle often consists of the initials of the personal name in the entry or the initials and an integer when the initials are not unique.

The connection to the NICNAME program is closed as soon as display of the matching entries or a message indicating the lack of matching entries is completed.

The NICNAME directory database contains about 10,000 entries. Individuals are responsible for entering and updating their own entries. To add, delete, or modify an entry, an individual sends a message containing the desired update via electronic mail to NIC@SRI-NIC. One of several individuals in charge of maintaining the NICNAME database enters the new information into the database using an update program that then rebuilds only the part of the database that was affected by the update.

PHONE

PHONE is a directory system that was designed for use within IBM. By typing "PHONE" followed by a string of characters, users can "look up" information about IBM employees who work at any of approximately 100 sites across the United States. An explanation about how to use the system is available by typing "PHONE ?".

The online phone book information at each site is organized into directories, with one directory for each IBM site, and users may specify in which of these directories they would like to search. Users also may specify that their private nickname files be searched. If a directory or nickname file is not specified, the directory for the local site is searched. Users may search for personal names, phone numbers, or computer login names. Typing "*" anywhere in the search string tells the program to match any character or characters in that position in the string. A blank is used to separate last name, first name, and middle name information in the search string. If a comma is typed after one of the parts of the name, the program searches for entries containing the string in the appropriate field exactly as typed. Otherwise, the program searches for entries containing a string in the appropriate field that begins with the typed characters. In other words, a comma is used to designate an exact match scheme instead of a prefix match scheme.

Entries matching the user's search string are displayed by screenfuls in a one line per entry format when possible. To display the next screenful of entries, the user issues a MORE command.[14]

Each IBM site is responsible for maintaining information about its employees. When the information at a site is updated, an individual at the site sends a new copy of the information file to the IBM site at Yorktown Heights, New York, where it is changed to the PHONE directory database format and then sent out to all the IBM sites. Thus, each site providing the PHONE service has a copy of the entire directory database, which contains approximately 27,000 entries.[15]

INQUIR

INQUIR is a directory service for M.I.T. computer users. It runs on several of the M.I.T. computers and provides information about individuals who have accounts on those computers. The directory service is invoked by typing "whois" followed by a string of characters. Entries in the directory database that contain a last name or a login name matching the typed string are displayed on the terminal screen, with each entry occupying several lines of the screen. The connection to the directory database is closed as soon as the display of matching entries or a statement indicating that there are no matching entries is completed. Thus, there is no need to exit the directory system.

Individuals with accounts on the computers that maintain an INQUIR directory database are responsible for entering and updating their own directory entries. To facilitate this process, access to the directory database is unrestricted. The commands for entering directory information are easy to use. An individual types "INQUIR" followed by a carriage return, then types one of several available

---

[14]How the command is issued is operating system dependent.

[15]Peter Capek, IBM Yorktown Heights, personal communication.

commands followed by a login name. The available commands are: NEW for creating a new entry, MODIFY for changing an entry, SHOW for displaying an entry, and EXIT for exiting the program. After typing NEW or MODIFY, one may enter such information as name, nickname, project, home address and phone, work address and phone, and birthday. Typing a question mark at any time causes a list of current options to be displayed.

In one version of INQUIR, the directory database contains entries only for individuals who have accounts on the machine providing the directory service. One such INQUIR database contains approximately 500 entries. Updates submitted to this INQUIR system are immediately added to the directory database and, thus, are immediately visible. In a second version of INQUIR, the directory database contains entries for individuals who have accounts on any of several machines. Each machine provides the INQUIR directory service and has a copy of the directory database, which contains approximately 2000 entries. Updates submitted to any of the machines running this version of INQUIR are mailed to the other machines running this version in order that all the databases contain the same information. Update programs on each machine then create new copies of the database. Thus, updates submitted to this INQUIR system are not immediately visible.

Bell Laboratories Experimental System

An experimental directory assistance system at Bell Laboratories in Holmdel, New Jersey, was designed to give feedback about the number of records in the database that match what the user has typed so far. The scheme used to match records is a phonetic matching scheme similar to the Soundex[16] algorithm. The user enters information such as personal name, department, building address, or phone number, and after each character typed, the system reports by means of a superimposed coding scheme the maximum number of database entries that match what has been

---

[16]Odel, Margaret K. and Russell, Robert C. *U.S. Patents 1261167* (1918), *1435663* (1922). Cited by [26].

typed so far [45]. The user can tell the system to "print" the entries on the screen by typing "p", output them to a file by typing "o", or count the actual number of matching records by typing "c". The system "remembers" the previous query so that queries can be combined by means of "AND" and "OR" operations. The database contains about 27,000 entries and currently is updated manually by editing the database file. The system is not widely used because it is not easily accessed: users must locate a terminal, log into one of the Bell computers, and start up the directory system. In addition, the database often contains out-of-date information.[17]

Comparison to DIRSYS

DIRSYS, like many of the surveyed directory systems, was designed to be used by individuals with widely varying computer skills and levels of experience in using the directory system. Unlike most of the surveyed systems, however, DIRSYS provides a tutorial and a detailed level of help in addition to the brief description of the available commands that most of the other systems provide.[18] Moreover, DIRSYS allows the user to get help without interrupting the immediate task. It must be noted, however, that if only a search command is available as in, for example, NICNAME, a more detailed level of help may not be needed.

In addition, since DIRSYS is modeled after a paper phone book, a screenful of information is comparable to a "page" in the electronic phone book. The surveyed systems do not necessarily display pages from the electronic phone book. Instead, they display a set of entries that match a typed string. In response to "whois lee", for example, the CSNET Name Server will display all entries that contain the string "lee" in any field, not just the last name field.

With DIRSYS, as well as the French, the New England Bell, the experimental Bell, and

---

[17] John Beyer, Bell Laboratories, personal communication.

[18] The CSNET Name Server is the exception among the surveyed systems; it also has two levels of help.

PHONE directory systems, a central authority is in charge of gathering and updating the information in the directory. In addition, DIRSYS provides users with a command for submitting an update request, and the update requests are sent to a manager before being incorporated into the database. The CSNET Name Server and INQUIR allow users to add, modify, and delete their directory entries without the intervention of a manager.

Finally, DIRSYS contains directory information for members of the M.I.T. community and is maintained on a central computer. All of the surveyed systems, except INQUIR, cross institutional or regional boundaries in terms of the information that they contain. They may not be maintained on computers in different locations, however. The CSNET Name Server and NICNAME, for example, contain information for individuals and institutions throughout the United States and are maintained on central machines. In contrast, PHONE contains information for individuals associated with one company, IBM, but the system is maintained at many sites.

1.3 Thesis Organization

Chapter 1 has presented an overview of this research and a survey of related work. Chapter 2 raises questions upon which this research was focused; the subsequent chapters seek to answer those questions. Chapter 3 describes the system, and chapter 4 presents a preliminary evaluation of it. Chapter 5 presents conclusions and examines questions that are topics for future research. Appendix A contains definitions of terms used in this thesis, and Appendix B contains performance graphs (which are discussed in Chapter 4).

Chapter Two

Operational and Design Issues

There are many issues to be resolved when building an online directory assistance system. Some of the issues, termed *operational issues*, were resolved on the basis of what seemed practical and organizationally efficient. These issues include acquiring directory information, identifying how to maintain and update the information, and identifying the users of the system. Other issues, termed *design issues*, became the focus of this research and include questions about the design of the interface, the database, and the update mechanism.

2.1 Operational Issues

Acquiring Directory Information

Who should be listed in the directory? The answer to this question is an organizational decision. In a business setting, the question of who should be included in a company-wide directory may not be difficult; the group of employees in a company is usually clearly defined. At M.I.T., however, there is a "soft" boundary between those who are associated with the Institute and those who are not. There are numerous and varied affiliations held with M.I.T. For example, individuals who are not members of the M.I.T. student body, faculty, or staff may be allowed guest accounts on M.I.T. computers. Some former members of the student body, faculty, and staff may maintain ties with the M.I.T. community after leaving the Institute, e.g., computer accounts or consulting with colleagues.

In making a decision about membership in the online directory assistance system, it was assumed that students and staff members listed in the M.I.T. paper phone books would be listed in the electronic phone book as well. Since the registrar's office

maintains records of student enrollment and the personnel office maintains them for faculty and staff appointments, and both offices provide information for the printed phone books, it was appropriate that these offices be a source of information for the electronic phone book.

Additional information, specifically electronic mail address information, was obtained from various M.I.T. computers. The electronic mail addresses were then added to the appropriate records of phone book information that had been obtained from the registrar's and personnel offices. If a person had electronic mail addresses on several M.I.T. computers, all the addresses were included in the directory and listed when a user viewed that record using DIRSYS. Thus, each entry in the electronic phone book contains the information available in the M.I.T. phone book[19] and electronic mail address information. When DIRSYS ceases to be a research tool, a method will be needed for asking people which information they would like listed in the online directory. For the time being, at the beginning of the fall term people are given the choice of not being included in this experimental online directory system by filling in a form that is distributed by the registrar's and personnel offices. Currently there is no proposed scheme for asking people which, if any, electronic mail address they would like listed in the online directory. The verification of electronic mail address information is organization dependent and can be done for the M.I.T. community if the registrar's and personnel offices ask for such information at the same time that they collect other phone book information.

Maintaining and Updating Directory Information

It was assumed that the system would be maintained on a computer dedicated to running the system in order to avoid complications and delays on a large time-sharing system that provides other services. In addition, having the entire

---

[19]This information includes M.I.T. address and phone number, home address and phone number, dorm phone number for students, department, and title (for faculty and staff members) or graduating year (for students).

system reside on one computer avoids problems associated with distributed databases. (Maintaining the system on more than one computer is discussed in Section 5.2.)

The system was designed on the assumption that it would be maintained by a manager, who would update the directory database according to policy issues formulated by a central authority and who would be responsible for maintaining the reliability of the information in the directory database. Putting a manager into the update scheme also might discourage unauthorized modification of the database since the commands that actually change the database would be available only to the manager.[20]

Requests to update information in the directory database could be sent to the DIRSYS manager by (1) the registrar's and personnel offices, (2) by the telecommunications operators (when someone informs the operators over the phone of new information), and (3) by persons who wish to send the new information directly to the manager.

The manager could assume that update information received from the registrar's or personnel offices was "official" and could be added immediately to the database. At present, there is no proposed way to verify the reliability and acceptability of the update information given to the telecommunication operators or sent directly to the manager. It is presumed that the DIRSYS manager would be able to devise validation procedures that would be effective. The presence of an authentication server perhaps would eliminate the need for validation by the manager when an individual sent an update request using DIRSYS from a machine on which the login name could be determined.

---

[20]DIRSYS is only as secure as the operating system on which it runs. No security features were introduced.

In addition to the updates that would be received on a daily basis,[21] academic institutions such as M.I.T. will have a large amount of new information to be added to the directory database, as well as a large amount of old information to remove, on a yearly schedule, e.g., in September. Eventually, the exchange of this information between the registrar's and personnel offices and the directory system manager could be automatic. It may be assumed at this stage that a new database would be built in the same way that the first one was built, i.e., based on current directory information obtained from the registrar's and personnel offices collected, say, at the beginning of the fall semester in September. An alternate method would be to merge the new information into the old database.

DIRSYS Users

Several classes of DIRSYS users were identified. The DIRSYS manager would not necessarily be an experienced computer user, but would be an experienced, routine DIRSYS user after initial training. The M.I.T. telecommunications operators would use the system routinely to provide directory assistance service for people who called in requests for information. They would not necessarily be experienced computer users, but would be experienced DIRSYS users after initial training. There would be two classes of infrequent users: those who are members of the M.I.T. community and those who are not. The infrequent user would be an inexperienced DIRSYS user, but could be either an experienced or inexperienced computer user.

DIRSYS users could access the system using typewriter-like keyboards and terminals connected directly to the DIRSYS machine and distributed throughout the M.I.T. community. Alternately, they could access the system from other terminals by means of a network connection to the DIRSYS machine. (An interface that employs a mouse is described in Section 5.2.)

---

[21] The number of daily updates probably would be small.

2.2 Design Issues

The Interface

Questions with respect to the interface, i.e., how the user interacts with the system, are as follows: Should the interface be *incremental*, i.e., should the screen be updated after each character that the user types? Or should it be *non-incremental*, i.e., does the user type a complete name and then hit an <ENTER> key to tell the system to start searching for that name? Should the user interact with the system by typing commands or by selecting commands from a menu? Perhaps learning aids could be designed to guide inexperienced users without encumbering experienced ones. Also, perhaps the system could uniquely identify the record of information for which the user was searching. How, then, might the system deal with nicknames and aliases, middle names and initials, as well as misspelled names? Should the system allow users to search on fields other than full name, i.e., could someone type a department name and then be shown a list of all people in that department? How might the same interface be used to send update information to the DIRSYS manager? Finally, how might the system enforce the M.I.T. privacy policy, which currently states that the phone book is not to be made publicly available, especially for commercial purposes? In other words, how could the system make reconstruction of the phone book difficult for someone outside M.I.T.?

In exploring answers to these questions, it was discovered that one interface design could not meet all the requirements. It was decided to experiment with an incremental interface to discover ways it might be used easily. The user could type only as much of a name as was required to find the desired entry. Ideally, the user would be able to see the desired entry on the screen after typing only a few characters. The incremental interface also had the potential of closely modeling the way people scan paper phone books. In addition, it was decided to experiment with designing an interface that employed simple typed commands rather than menus of commands. It was hoped that novices could learn to use the system as quickly as with menus and that the system, once learned, would be easier and faster to use than

a system with menus. With the incremental interface, perhaps the user could deal more quickly and efficiently with nicknames than the system could and, moreover, the identification of a unique phone book entry could be left up to the user. The incremental interface, however, could not deal effectively with misspellings and searching on fields other than name (where the last name is included) since its design would be organized around searching for information using alphabetized names. Thus, a second interface would be needed. The incremental interface could be used to send update information to the manager. People in the registrar's and personnel offices, the telecommunication operators, and people wanting to change their own directory information could use the same interface for submitting update information. Finally, the incremental interface would not address privacy issues well enough for it to be used by people outside the M.I.T. community because there would be no way to limit the amount of information that appears on the screen. A separate interface, then, would have to be implemented for use outside M.I.T. (The incremental interface is described in detail in Section 3.1. Additions to this interface and an interface for use outside M.I.T. are discussed in Section 5.2.)

The Database

Questions about the database design included: What is a simple database structure that will allow fast system response? What kind and how many levels of indexing are needed?

It was decided to set up the database as a sequential file of records with three levels of indexing. This organization would facilitate the random accessing needed when a user types a name and the sequential accessing needed for displaying consecutive records of information on the screen. The main index was to take advantage of the fact that the database was to be ordered alphabetically by last name by having names, or parts of names, from the database as its index entries. (The organization of the database and its indexes is described in detail in Section 3.2.)

Update Mechanism

Finally, questions about the scheme for keeping the database up-to-date needed to be addressed. How could the database be updated without changing the database that DIRSYS was using, i.e., so that the system could remain reliable? How could the database be updated simply without degrading system performance? How often should updates be incorporated into the database?

It was decided that in order not to "disturb" the database that DIRSYS was using, the updates should be kept in a file separate from the database and that they should be added to a copy of the database. In this way, a copy of the old database also could be kept intact in case something went wrong with the update process. Since the update program would need to copy the database, it seemed expedient that it copy the database and update it at the same time. Thus, the database and indexes would be completely rebuilt when updates were added to the database. In addition, rebuilding the database would be much simpler than having the daemon only rebuild parts of the database, and the daemon could use the same build procedure that was used to build the database and indexes the first time.

It seemed appropriate to rebuild the database at night rather than during the day when the number of users might be large, thus reducing the possibility of degrading the system response time. The rebuild process could be automatic. An update program could check to see if the database needed to be rebuilt, and if so, rebuild it. The manager could switch DIRSYS to the new database the next morning after having inspected the new database to make sure that it had been rebuilt properly.

It is generally expected that once a paper phone book is printed, changes will not appear until the next issue. New directory information may not be available in print for as long as a year. With an online directory, however, there is no printing of a phone book involved; changing a file on a computer is required. Thus it may be feasible to update the directory database frequently. Updates made more often than once a day would not be necessary, however, since a maximum wait of 24 hours

before new information would be visible would not seem an excessive delay, especially when the update validation step takes a few days. (The update procedure is discussed in more detail in Section 3.3.)

Chapter Three

The System

DIRSYS, implemented in the CLU programming language [30] on a DEC-SYSTEM 20 and a VAX 11/750,[22] has three parts: the interface, the database, and the update file. This chapter describes each of these parts.

3.1 The Interface

In this section, principles that were followed in designing the interface are presented, followed by a description of the interface from the user's point of view. The section ends with a description from the designer's point of view of the interface features that illustrate the design principles.

3.1.1 Design Principles

As mentioned earlier, the primary goal for DIRSYS was to design a system that could be used by inexperienced, experienced, and occasional users. Therefore, the DIRSYS interface, which is the part of the system through which users interact with DIRSYS, needed to be designed to reflect the needs of all classes of users. It needed to be easy to learn to use and easy to use once it had been learned.

Several principles guided the interface design and are discussed below. Features of the DIRSYS interface that illustrate each of these principles will be pointed out in the interface description and in the section following the description.

---

[22] The DEC-SYSTEM 20 runs the TOPS-20 operaing system; the VAX 11/750 runs the UNIX operating system.

1. **User's Model**: Users often perform best when they have a clear conceptual model of the system that they are using [7, 16, 28, 38]. The model enables the users to understand what the program is doing and to anticipate the effect of their actions. It also enables them to develop their own strategies for using the system. In order for the user's model to be intuitive and easy to learn, it has been suggested that the system, and specifically the interface, be based on concepts that are familiar to the user [38].

2. **Minimal Command Set**: Users may be overburdened by systems that contain more features than are needed for completing a particular task. They may be tempted to learn all features even though some of them are not needed for the task. In addition, redundant forms of the same operation may cause users to be confused and have difficulty remembering the choice of ways to perform an action [28]. There is evidence from decision theory that suggests that user performance is improved when the number of alternatives with which a user is confronted has been minimized [13, 33]. In addition, there is experimental evidence to support the assertion that users perform best with the smallest possible command set that permits completion of a given task [2]. In light of this assertion, extraneous features, i.e., those unrelated to the task, and redundant features, i.e., duplicate forms of commands, should be avoided. It should be noted that these assertions probably would not be substantiated for a system designed only for experienced computer users.

3. **Simple Command Structure**: Evidence suggests that for commands to be easy to learn and easy to remember they should have a simple structure and their names should be based on English (or German, French, etc.) phrases composed of familiar descriptive words [7, 17, 28, 38]. For systems that employ typewriter-like keyboards, command abbreviations aid in rapid, correct input of commands [28]. If the abbreviation scheme is consistent and simple, it often contributes to better user performance by reducing the effort in memorizing commands and in reducing the number of errors in entering abbreviated commands [23]. Experimental data suggests that the most easily remembered abbreviation scheme is first letter abbreviation.[23] First letters are also fast and easy to type. To use this scheme, however, the command set must not be large or command names may become obscure in an attempt to keep first letter abbreviations unique.

---

[23]Freedman, J. L., and Landaur, T. K. Retrieval of Long-Term Memory: Tip-of-the-Tongue Phenomenon. *Psychological Science*, August, 1966. Cited by [28].

4. **Consistent Command Interpretation**: Evidence suggests that to decrease users' confusion, commands should not have different meanings in different contexts [17, 28, 38]. Instead, they should consistently perform the same actions. Some systems, for example, may interpret the letter Q as quit in one context and as text in another. To avoid user confusion and decrease the number of user errors due to not knowing which command meanings are associated with the current context, systems should not interpret commands differently in different contexts.

5. **Help Facility**: The command for getting help should be simple and concise. The user should be able to get help from the system at any point in the user session without interrupting the immediate task, and the assistance provided should be specific to the user's current context [43, 44, 49]. Users differ in the amount of help that they need or want. A novice may want a detailed description of a certain command, while an occasional user may want a sentence or two, and an experienced user may want only a brief explanation. Thus, the system should provide different levels of help.

6. **Feedback**: Users should receive feedback from the system so that they know what the system is doing. Feedback includes echoing of typed characters, indication of a selected object, acknowledgment of receipt of a command, and explanatory messages [14, 38]. The explanatory messages should be displayed in a conspicuous place on the screen, but should not interfere with the user's text. They should be concise, polite, understandable, and informative [28, 43, 44, 47, 49]. Examples of explanatory messages include notifying the user that the system is still working on a request (if a particular command execution is slow) and notifying the user when he has made an error (since continued commands may be invalidated by the previous error). If the message is the result of a user error, it should appear immediately after an error so that the user does not have to mentally reconstruct what he did [14, 38]. The error message should not only indicate that there was an error, but also should explain the error and indicate what the user may do next. Error messages should be phrased positively and should avoid implication that the user is at fault [28, 47].

   In addition, the user should be able to give the system maintainer feedback. Comments about difficulties are useful in improving and maintaining the system.

7. **Error Handling**: The system should be resilient against a variety of user errors. Thus, it should be virtually impossible for the system to terminate abnormally, i.e., the system should be *robust* [7, 25, 55]. If a system error should occur, a message explaining the error should be sent to the system maintainer. A message should be sent to the user informing him that a system error has occurred, but not detailing the error. The underlying aspects of the computer should be invisible to the user, and the user should not be distracted by information intended for the maintainer [44].

3.1.2 What the Interface Looks Like

In this section, the interface is described from the user's point of view. Its features are illustrated by describing how the user interacts with the system.

Starting DIRSYS

When DIRSYS starts up, the first screen, the *herald*, shown in Figure 3-1, contains a brief description of what the system does, how to start using it, and a list of the commands needed by the first-time user, along with the functions of those commands. It also gives an address to which users may send comments.[24]

After the herald has been displayed, the user has two options: he may begin searching for a name, or he may get additional instruction on how to use DIRSYS by starting a tutorial. (Searching for a name is discussed in the next subsection; the tutorial is discussed in the subsection entitled "Getting Help.")

Looking Up a Name

Once the herald has been displayed, issuing the exit screen command, CTRL-E[25], causes the herald to be replaced by a screen that is divided into four rectangular areas, called *windows*: the label window, the directory data window, the status window, and the echo area. This screen is called the *search screen* and is shown in Figure 3-2.

The top line of the screen is the *label window*. It contains information about the window that is located just below it. For example, in the screen that appears immediately after the herald, the label window contains headings for the columns of information that appear in the directory data window.

---

[24]Currently, the address is an electronic mail address. A better alternative would be to have a command in DIRSYS for sending comments directly to the manager, since users may not have access to an electronic mail system.

[25]The notations "CTRL-E" and "↑E" represent hitting the key labeled "E" while holding down the key labeled "CTRL".

```
                D I R S Y S  --  MIT Directory Assistance System

DIRSYS is an electronic MIT phone book.  A brief explanation of how to use it
is given below.  If you have never used DIRSYS before or want to review how to
use it, you may want to start the tutorial, by typing ↑T. (This notation means
hit the key labeled "T" while holding down the key labeled "CTRL".)

** To start DIRSYS, hit ↑E. **
Then to start searching for a name, begin typing the name in the form:

                        Last name, First name, Middle name

Other useful commands:
       ?       HELP (lists available commands and their functions).
      ↑Q       Quit the program.
      ↑N       go to Next line.
      ↑P       go to Previous line.
      ↑F       go Forward one screen.
      ↑B       go Backward one screen.
      <DEL>    Delete one character from what you have typed.
      ↑K       Kill the search and start over.
      ↑S       display more information about entries (Switch format).

You may send comments via electronic mail to DIRSYS at MIT-XX.
```

Figure 3-1: Herald Screen

The bottom line of the screen is the *echo area*, and the characters that the user types appear there. It contains the following prompt at the beginning of the line: "Name (last, first, middle): ".

The two lines above the echo area form the *status window*. The top line of the status window is used to send messages to the user. For example, immediately after the herald disappears, DIRSYS sends the user a message saying that it is getting the first screen of entries and to please wait. The bottom line of the status window contains a list of available commands.

The *directory data window*, or *data window*, takes up the largest area of the screen. It is located in the center of the screen between the label and status windows and contains the "pages" of the electronic phone book. Each line on a page gives information associated with a person who is listed in the directory. This line of

```
Name.....................Hm Phone.....MIT ext....Dept.................Status

ALLEN, LARRY, W ..... 646-3080 ... 3-6020 ... LAB FOR COMPUTER SCI ..STAF
baldwin, robert, w        494-8490     3-6020     elec eng & comp sci    g
berlin, stephen, t                     3-1448     lab for computer sci   staf
bridgham, david, a        255-6683                                       1985
comfort, sarah                         3-6002     lab for computer sci   staf
cooper, geoffrey, h                    3-6006     elec eng & comp sci    g
corbato, fernando                      3-6001     elec eng & comp sci    fac
estrin, deborah           497-9491     3-6005     elec eng & comp sci    g
feldmeier, david, c       255-9540                elec eng & comp sci    1984
gifford, david k                       3-6039     elec eng & comp sci    fac
gramlich, wayne, c        494-1076     3-6042     elec eng & comp sci    g
greenwald, michael, b     497-0472     3-6042     lab for computer sci   staf
harteneck, ralf           494-9833     3-6020     elec eng & comp sci    1984
hopkins, grace                         3-6042     elec eng & comp sci    1984
hornig, charles                        3-7788     elec eng & comp sci    1983

COMMANDS: ?, ↑T, ↑R, ↑N, ↑P, ↑F, ↑B, ↑S, <DEL>, ↑W, ↑K, ↑Q, ESC-M
Name (last,first,middle):
```

Figure 3-2: Sample Search Screen

information is called an *entry*. The one line per entry format allows the user to scan the pages quickly, since it allows many entries to be displayed per screen.

The user's attention is focused on one entry by *highlighting* that entry. DIRSYS highlights an entry by displaying the information in reverse video if the terminal has this capability and in uppercase letters, leaving other entries in lowercase. It also fills spaces between fields of information with periods.

To focus the user's attention in the center region of the screen, the highlight only travels between an upper and a lower margin in the data window.[26] Should the entry to be highlighted be above the upper margin, the data window will scroll backward, placing the highlighted entry at the lower margin. Similarly, when the highlighted entry appears below the bottom margin, the data window scrolls forward, placing the highlighted entry at the top margin. In this way, the highlighted entry always has at

[26] These margins are not visible on the screen.

least a margin's worth of entries above or below it for context. Consideration was given to keeping the highlight in the center of the screen, i.e., setting the margins equal to one-half of the data window height. In this scheme, the background text rather than the highlight would move when the highlighted entry changed. Thus, the entry of interest would always be in the center of the screen. It was found that the user had a difficult time focusing on the highlighted entry with the background constantly changing. As a compromise between letting the highlight appear anywhere on the screen, i.e., having no margins, and keeping the highlight in the center of the screen, i.e., having margins of one-half the data window height, the margins currently are set at one-fifth of the data window height.

DIRSYS provides the user with two ways for looking up a name. The user can type the name into the echo area, letting DIRSYS display the appropriate page, or he can use commands to scroll forward or backward through consecutive pages of the directory database.

As the user types characters into or deletes characters from the echo area, DIRSYS checks to see if the first matching entry is already on the screen. If it is, then the highlight is moved to that entry. If it is not, DIRSYS sends the message, "Searching for *x*, please wait...", where *x* is what the user has typed into the echo area. When the new page is displayed, the matching entry is highlighted and appears at the top margin. The process is repeated as the user continues to type or delete characters.[27]

When the last character is deleted from the echo area, the highlight does not move.[28] In a earlier version of the interface, the highlight returned to the beginning of the database when the echo area was empty. In this way, deleting characters from the echo area always had the same effect, that of searching for the closest matching

---

[27] As an optimization, DIRSYS may take several characters at a time before redisplaying the screen.

[28] The last character may be deleted with <DEL>, CTRL-W, or CTRL-K.

entry. The user could also get a sense of starting over. It was found, however, that when the user searches for several different names, the redisplay between searches is distracting. Redisplaying the first page of the directory database also seemed unnatural since one seldom goes back to the first page of a paper phone book when looking up several names.

If there is no entry in the directory database that matches what has been typed in the echo area, DIRSYS will inform the user and highlight the closest matching entry. In this way, the user can verify that no matching entry exists and perhaps will see an alternate entry. For example, if the user types the name "Bob Smith" and does not find it listed in the directory, he might notice an entry for "Robert Smith" and decide that that one is the desired entry.[29]

Instead of typing the name into the echo area, the user can issue commands to move the highlight forward or backward through the directory. Commands are available for moving the highlight to the previous entry, the next entry, the previous page, or the next page. When one of these commands is given, the echo area does not change to match the newly highlighted entry; it contains only the characters that the user has typed, if any, because it was judged that having the program change the contents of the echo area in order to match the highlighted entry would be disconcerting to the user.[30] Subsequent typing into the echo area will cause the highlight to be positioned as if it had not been moved manually. Thus, typing into the echo area takes precedence over manually moving the highlight. An alternative would be to indicate to the user which characters in the echo area match the currently highlighted entry by displaying the matching characters in the entry of interest in high intensity highlighting and the rest of the entry in low intensity highlighting.

Should the user want to see more information than is displayed in the one line per

---

[29]More sophisticated matching schemes could find nicknames. This idea is discussed further in Section 5.2.

[30]This idea needs to be tested.

```
EXPANDED DISPLAY                                    (↑S switches format)

Name:  corbato, fernando, j     (corbato@mit-xx)
Status:    professor, elec eng & comp sci
Work addr:  ne43-5xx; x3-6001
Home addr:  home; 527-6204

Name:  ESTRIN, DEBORAH, L     (estrin@mit-xx)
Status: GRADUATE, ELEC ENG & COMP SCI
Work Addr:  NE43-508; x3-6005
Home Addr:  CAMBRIDGE, MA; 253-6006

Name: feldmeier, david, c
Status:  1984, elec eng
Work addr:  ne43-504
Home addr:  home; 255-9540

Name: gifford, david, k
Status: assistant professor, elec eng & comp sci
Work addr:  ne43-507; x3-6039
Home addr:  home; 123-4567


COMMANDS: ?, ↑T, ↑R, ↑N, ↑P, ↑F, ↑B, ↑S, <DEL>, ↑W, ↑K, ↑Q, ESC-M
Name (last, first, middle):
```

Figure 3-3: Sample Search Screen in Expanded Format

entry format, or *compressed format*, a command (CTRL-S) can be given to switch the entries in the directory data window into *expanded format*, shown in Figure 3-3. The system commands and the windows are the same as when entries are in compressed format. The label window indicates that the entries are in expanded format and reminds the user how to switch formats. The echo area functions in the same way as with compressed format. The top line of the status window still contains messages to the user, and the bottom line of the window still contains a list of available commands. In expanded format, the directory data window contains fewer records of information on the screen at one time, since each entry takes up more space. Highlighting an entry in expanded format means displaying it in capital letters, with each line of the entry in reverse video if the terminal has this capability. The same command, CTRL-S, returns the directory data window to compressed format. In this way, the user has to remember only one switch format command.

Getting Help

The user can ask for help by typing CTRL-T or "?". The CTRL-T command starts a tutorial which explains in detail how to use DIRSYS. It was designed to help the novice understand what the commands do and how to use them. It must be asked for explicitly with a command so that the user can get the detailed instruction only if he wants it. The experienced computer user or experienced DIRSYS user probably would not need to read it.

The screen that appears when starting the tutorial contains the label window, the status window, and the *tutorial window*. (See Figure 3-4.) The echo area is now blank since the user cannot add characters to it while in the tutorial. The label window gives the name of the currently displayed tutorial section and reminds the user to type CTRL-E to exit the tutorial. The top line of the status window tells the user that he is in the tutorial and gives the names of the previous and next tutorial sections. The bottom line of the status window lists the commands that can be issued while in the tutorial.

The tutorial window replaces the directory data window and contains the text of the tutorial. The text is organized into seven sections: What is DIRSYS?, Getting Started, Looking Up a Name, Expanded Format, Getting Help, and Requesting an Update. The section titles are listed at the beginning of the tutorial. The user moves through the text by using CTRL-F to move forward a page and CTRL-B to move backward a page.

The help facility that is available using the "?" command was designed to aid both inexperienced and experienced users and provides help only when and where required. It has two levels, the first of which reminds the user of the available commands and command functions. The second level gives a more detailed explanation of a specified command. Thus, the help facility serves as a learning tool for first-time users and as a reminder for experienced, but infrequent users.

```
TUTORIAL: Getting Help                                  (↑E exits tutorial)

GETTING HELP

The help facility, available by typing "?", has two levels.  The first level
reminds you of the available commands and their functions.  The second level
gives a more detailed explanation of a specified command.

The first level help window disappears when you type any letter or one of the
commands listed in the help window.  The window that was on the screen before
you typed "?" will be restored.  If a letter was typed, it will be added to
the screen in the appropriate place.  If a command was typed, it will be
executed in the restored window.

Once the first level help window is on the screen, you may request more help
on a command by typing another "?"  followed by that command.  A second
level help window will appear that contains a detailed explanation of the
command that you typed.  You may return to the first level by typing CTRL-E.


TUTORIAL SECTIONS:  Prev -- Looking Up a Name; Next -- Requesting an Update
COMMANDS: ↑F (next page), ↑B (prev page), ↑E (exit), ↑R (redisplay), ↑Q (quit)
```

Figure 3-4: Sample Tutorial Screen

The *help window* that appears after typing a "?" is best thought of as a *pop-up window* in that it appears when the user issues the help command and disappears when the user begins typing again. It covers up the label window and the top part of the directory data window.[31] The echo area and status window remain on the screen.

The content of the help window differs slightly depending on which window was on the screen before the help window appeared. Each help window contains a list of currently available commands and their functions and brief instructions about how to get more help on any of the commands listed. Any other options available to the user are also listed. If the user types a "?" from the directory data window, for example, a brief explanation of how to start, or continue, searching for a name also appears in the help window. (See Figure 3-5.)

---

[31] The fraction of the data window that is covered up depends on the screen size and the amount of text in the help window. In some cases, the help window completely hides the data window.

```
AVAILABLE COMMANDS -- type one of the following:

    ↑E      Exit current screen
    ↑R      Redisplay current screen
 (Any of the following commands may be executed from this help window. The
  data window will reappear, and the command will be executed there.)
    ↑T      start the Tutorial
    ↑N      move to Next line
    ↑P      move to Previous line
    ↑F      move Forward a screen
    ↑B      move Backward a screen
    ↑S      Switch to expanded format
    <DEL>   Delete a character (use the <DEL> key)
    ↑W      delete a Word backwards
    ↑K      Kill search and start over
    ↑Q      Quit the program
    <ESC>-M request that your entry in the directory be Modified

 or to get more help on a command, type another "?", then that command
 or to search the directory, start typing a name
--------------------------------------------------------------------------

COMMANDS: ↑E,↑R,↑T,↑N,↑P,↑F,↑B,↑S,<DEL>,↑W,↑K,↑Q,<ESC-M>,? followed by command
Name (last, first, middle):
```

Figure 3-5: Sample First Level Help Window

The user may execute one of the commands or options listed in the help window without explicitly exiting the help window first. As soon as the user types a command or character (except "?"), the help window disappears, the previous window reappears, and the command is executed in the restored window or the character is added to the screen in the appropriate location. If, for example, the user types a "?" from the directory data window and then types a character once the help window appears, the directory data window will reappear, and the character will appear in the echo area.

If the user requests more help on a command, by typing "?" followed by the command, then a second level help window appears, replacing the first one. (See Figure 3-6.) The echo area is now blank since the user cannot add characters to it while in the second level of help. The new help window contains a detailed explanation of how to issue the specified command and what that command does. It

```
DETAILED EXPLANATION OF ↑N                              (↑E exits this help)

↑N, issued by hitting the "N" key while holding down the "CTRL" key, causes
the highlight to be moved down one line.  (The next screen will be displayed
if the highlight is near the bottom of the screen.)


Current Options:
     (1) type ↑E to exit and return to the first level of help,
     (2) type ↑R to redisplay this screen
     (3) type ↑Q to quit DIRSYS

-------------------------------------------------------------------------
comfort, sarah                          3-6002     lab for computer sci   staf
cooper, geoffrey, h                     3-6006     elec eng & comp sci    g
corbato, fernando                       3-6001     elec eng & comp sci    fac
estrin, deborah             497-9491    3-6005     elec eng & comp sci    g
gifford, david k                        3-6039     elec eng & comp sci    fac
gramlich, wayne, c          494-1076    3-6042     elec eng & comp sci    g
greenwald, michael, b       497-0472    3-6042     lab for computer sci   staf
harteneck, ralf             494-9833    3-6020     elec eng & comp sci    1984
hopkins, grace                          3-6042     elec eng & comp sci    1984

COMMANDS: ↑E, ↑R, ↑Q
```

Figure 3-6: Sample Second Level Help Window

also lists the user's current options. Should the user type only a "?" from the first help window, the prompt, "Command character:", will appear in the echo area. When the user then types a command, it is displayed after the command prompt.

Consideration was given to providing more options at the second help level, including allowing the user to ask for help on another command from the second help level and allowing the user to exit the second level help window without having to return to the first help level. With the latter option, the user would be able to execute a command from the second level help window in the same manner as from the first level help window. The user could type CTRL-N from the second help level, for example, and the directory data window would reappear with the highlight moved to the next entry. Since these options might seem confusing and unnecessarily complicated to a user unfamiliar with DIRSYS, they were not provided.[32] The user can quit DIRSYS from the second level help window, however.

[32]Those familiar with DIRSYS might like the extra options, but probably would not use the second level of help very often.

Requesting an Update

The user can request that his entry in the directory database be modified.[33] (How the updates are added to the database is discussed in Section 3.2.) Typing ESC-M causes a screen to appear that contains the label window, the status window, and the *update request window.* (See Figure 3-7.) The echo area is now blank because typed characters will appear in the update request window rather than in the echo area. The label window indicates that a modification request is being submitted and reminds the user how to submit the request and exit the window. The top line of the status window still contains messages to the user, and the bottom line of the status window lists the commands that can be issued while the update window is on the screen.

The update request window replaces the directory data window and contains a brief explanation of how to request an update and a "form" that contains the user's directory information. The user is instructed to erase the information that is no longer valid and replace it with the new information. The form can be edited by using CTRL-N or a carriage return[34] to move to the next line, CTRL-P to move to the previous line, <DEL> to delete a character, CTRL-W to delete a word, and CTRL-K to "kill" a line of information. He is then instructed to type CTRL-U to submit the update request and CTRL-E to exit the update request window. The user also is informed that he may exit the window without submitting a request by typing only CTRL-E. Finally, he is warned that the update requests are reviewed by the DIRSYS manager and that usual policy forbids changes to any record other than his own.

When an update request for a record has been submitted, the proposed changes for

---

[33] One of the duties of the DIRSYS manager is to insure that a person can modify only his own record of information. The presence of an authentication server perhaps would relieve the manager of this duty in the case of an individual sending an update request using DIRSYS from a machine on which the login name could be determined.

[34] Carriage return was added since it is natural for people who are used to typing on typewriters or typewriter-like keyboards to hit a carriage return to move to the next line.

```
MODIFICATION REQUEST                    (↑U submits request, ↑E exits)

To request that the following record of information be modified, erase old
information and type in new information.  Then type ↑U to submit the request
and ↑E to exit the screen.  You may exit the screen without submitting the
update request (by typing only ↑E). (Use <DEL> to delete a character, ↑W to
delete a word, ↑K to kill a line, ↑N or <RETURN> to go to next line, ↑P to go
to previous line.)  Update requests are reviewed by the DIRSYS manager.
Usual policy forbids changes to any record other than your own.

   NAME (last, first, middle):  Smith, Larry, D
   HOME STREET ADDRESS:         1234 West End Dr.
   HOME CITY, STATE, ZIP:       Arlington, MA  02174
   HOME PHONE:                  648-9000
   WORK STREET ADDRESS:         MIT
   WORK CITY, STATE, ZIP:       Cambridge, MA  02139
   WORK PHONE:
   DEPARTMENT:                  elec eng & comp sci
   TITLE OR GRADUATING YEAR:    assistant professor
   ELECTRONIC MAIL ADDRESS:


COMMANDS: ↑N, ↑P, ↑E, ↑U, <DEL>, ↑K, ↑W, ↑Q, ↑R, ↑T, or ? for help
```

Figure 3-7: Sample Update Request Window

that record are visible the next time the record is displayed in an update request window. (See Figure 3-8.) In this way, a person will know what update information he has already submitted. The proposed new information is not shown in the directory data window until it has been validated by the manager. When it has been validated but not incorporated into the database yet, it will appear in place of the old information whenever the record is displayed because DIRSYS searches the update file for valid information before displaying a new page of directory entries.[35]

Consideration was given to allowing the user to submit addition and deletion requests as well as modification requests. Since individuals who are in the paper phone book do not decide whether or not they are included in that phone book, it was decided that DIRSYS should not allow users to decide whether or not they

[35] It is hoped that the number of update requests will be small so that this search will not increase system response time.

```
MODIFICATION REQUEST                    (↑U submits request, ↑E exits)

To request that the following record of information be modified, erase old
information and type in new information.  Then type ↑U to submit the request
and ↑E to exit the screen.  You may exit the screen without submitting the
update request (by typing only ↑E). (Use <DEL> to delete a character, ↑W to
delete a word, ↑K to kill a line, ↑N or <RETURN> to go to next line, ↑P to go
to previous line.)  Update requests are reviewed by the DIRSYS manager.
Usual policy forbids changes to any record other than your own.
Changes already submitted are in [ ].

   NAME (last, first, middle):  Smith, Larry, D
   HOME STREET ADDRESS:         1234 West End Dr.
   HOME CITY, STATE, ZIP:       Arlington, MA  02174
   HOME PHONE:                  648-9000
   WORK STREET ADDRESS:         MIT                      [ NE43-500, MIT ]
   WORK CITY, STATE, ZIP:       Cambridge, MA  02139
   WORK PHONE:                                           [ 3-6000 ]
   DEPARTMENT:                  elec eng & comp sci
   TITLE OR GRADUATING YEAR:    assistant professor
   ELECTRONIC MAIL ADDRESS:

COMMANDS: ↑N, ↑P, ↑E, ↑U, <DEL>, ↑K, ↑W, ↑Q, ↑R, ↑T, or ? for help
```

Figure 3-8: Sample Update Request Window Showing Proposed Changes

should be included in the electronic phone book. The policy covering who is added to or deleted from the paper and electronic phone books is decided by the registrar's and personnel offices. Perhaps addition and deletion request commands could be available for the DIRSYS manager to use.

Surveying and Marking Updates[36]

The commands for processing update requests, i.e., *surveying* and *marking* them, are part of a separate program that is available only to the DIRSYS manager. They cannot be invoked from DIRSYS. The interface to the program, however, is very similar to the DIRSYS interface. Most of the DIRSYS commands are used in this program and retain their semantics. In addition, other commands are available for

[36]Update commands for the manager are not implemented yet.

processing the update requests. A tutorial and help facility that are identical to those in DIRSYS are available, except that they also contain information on the manager's additional commands.

When an update request is made, DIRSYS sends the manager a message notifying him that a new request exists. The manager will then use the update program to view the new requests by *surveying* the update requests and will attempt to determine whether or not the request contains valid information, e.g., via letters or phone calls to the person whose record is listed in the request. When he has determined whether or not a request is valid, the manager will edit the request if necessary, *mark* the update request as *valid*, *invalid*, or *waiting*, and the update incorporation program, or *update daemon*, that runs at night will remove the update request from the list, log the information that the request contains, and either add the update to the database or not (depending on whether the request is valid or invalid).

The screen that appears when the survey requests command is given, shown in Figure 3-9, contains the label window, the status window, the echo area, and the *update survey window*. The label window contains column headings for the information given in the update survey window. The top line of the status window is used to send messages to the manager, and the bottom line of the status window lists the commands that can be issued while the update survey window is on the screen. The echo area is used for incrementally searching the list of update requests and functions in the same way that it does in DIRSYS.

The update survey window, located in the same position on the screen as the directory data window in DIRSYS, lists information for each update that has been requested and that has not been incorporated into the database yet. The information associated with an update request occupies one line of the screen, i.e., is in *compressed format*, and contains four items: the name in the record to be updated, whether the manager has seen the update request, the *update status*, or *status*, of the request, i.e., whether it is *valid*, *invalid*, or *waiting* for verification, and the date

```
Name.....................Seen.........Status.......Date Marked

ADAMS, MARY, W ..........  --  ....... waiting .....  --
baker, robert, m           --          waiting        --
clark, michael, d          --          waiting        --

baker, robert, m         6-10-83       waiting        --
wilson, nancy            6-10-83       waiting        --

anderson, rebecca        6-12-83       invalid      6-14-83
bern, stephen, t         6-10-83       invalid      6-15-83

graham, karen, s         6-10-83       valid        6-15-83
davis, james, a          6-10-83       valid        6-15-83
jackson, david, t        6-10-83       valid        6-15-83


COMMANDS: ?,↑N,↑P,↑F,↑B,<DEL>,↑K,↑W,↑S,↑R,↑Q,↑T,ES-S,ES-U,ES-V,ES-I,ES-W, ES-E
Name (last,first,middle):
```

Figure 3-9: Sample Update Survey Window

---

that the status was marked, if it is marked. The requests are displayed in the following order: those marked unseen are listed first, followed by those marked waiting, then by those marked invalid, and finally by those marked valid. All the unseen requests should be marked waiting, but if there are some unseen requests that are marked invalid or valid, the waiting requests are listed first, followed by the invalid requests, then by the valid ones. Within each of the four groups -- unseen, waiting, invalid, or valid -- requests are ordered alphabetically.

The manager can scan the requests in several ways. He can move the highlight manually as in the DIRSYS directory data window using the CTRL-N (next line), CTRL-P (previous line), CTRL-F (next page), or CTRL-B (previous page) commands. He can move around within the current group[37] by typing a name into the echo area. The update requests are searched incrementally for the matching name, and the highlight is moved to the line containing that name. He can move to the next or

---

[37] The current group is the group that contains the highlight.

previous group with ESC-N or ESC-P, respectively. Upon jumping to a new group, the contents of the echo area are used to locate and highlight the matching entry in that group. In this way, the manager can locate all requests for one person.[38] A blank echo area causes the first request in the group to be highlighted.

The manager marks a request as *seen* by typing ESC-S or *unseen* by typing ESC-U. The requests are marked unseen by default. The manager would leave a request marked unseen or mark a previously seen record as unseen if he wanted to be reminded of it. The update daemon would send a message notifying him of the unseen request, and when the requests were surveyed again, the request would be easy to locate since it would appear near the beginning of the list.

The manager also can mark an update request *valid* by typing ESC-V, *invalid* by typing ESC-I, or *waiting* for verification by typing ESC-W. The requests are marked waiting by default; therefore, the manager would only mark a request waiting if it had previously been marked valid or invalid. The status is immediately visible on the screen, and the requests are reordered to reflect the new status.

Should the manager want to view an update request in more detail, he issues the switch format command, CTRL-S, which functions in the same way that it does when issued from the directory data window in DIRSYS. The update requests are then displayed in *expanded format*.[39] (See Figure 3-10.)

In expanded format, the label window indicates that the update survey window is in expanded format and reminds the manager how to return to the compressed format (with CTRL-S). The echo area functions in the same way as with compressed format. The top line of the status window again is used to send messages to the user, and the bottom line of the status window again contains a list of available commands. The

---

[38] One person may have several update requests, each of which may have a different status.

[39] Depending on the screen size, some terminals may display only one request per screen.

```
UPDATE SURVEY WINDOW: Expanded Display                    (↑S switches format)

Name:  Smith, Larry, D.
Title & Dept:  assistant professor;  architecture
Work Addr & Phone:  MIT;  3-6000
Home Addr & Phone:  1234 West End Dr., Arlington, MA;  648-9000
Electronic Mail Addr:

NEW: WORK ADDR & PHONE:    NE43-500, MIT;  3-6000
     Electronic Mail Addr: LDS at MIT-XX

     submitted:  6-06-83          by:  Smith, Larry, D
     marked:  seen                by:  Davis, Alice          on:  6-10-83
     marked:  valid               by:  Davis, Alice          on:  6-15-83

Comments: telephoned at 3-6000 on 6-15-83; ok


COMMANDS: ?,↑N,↑P,↑F,↑B,↑S,<DEL>,↑K,↑W,↑R,↑Q,↑T,ES-S,ES-U,ES-V,ES-I,ES-W,ES-E
Name (last, first, middle):
```

Figure 3-10: Sample Update Survey Window in Expanded Format

---

expanded request contains: the current database record, the new information, the date that the request was submitted and by whom, whether the request has been seen, and if so, the date that it was seen and by whom, the status of the request, the date that the status was marked and by whom, and comments about the request, such as when and how the request was verified.

The manager can scan the requests in expanded format using the echo area or using CTRL-N, CTRL-P, CTRL-F, and CTRL-B as in compressed format. If only one entry can be displayed per screen, CTRL-N and CTRL-P will function in the same way as CTRL-F and CTRL-B, respectively. Also as in compressed format, the ESC-N and ESC-P commands are available for moving between groups.

Finally, the manager can edit a request if necessary. Typing ESC-E causes the update survey window to be replaced with an *update edit window*, which contains the highlighted update request in expanded format. (See Figure 3-11.) The label and

```
UPDATE EDIT WINDOW                                    (↑E exits this window)

Move between lines with CTRL-N, CTRL-P; edit with <DEL>, CTRL-W, CTRL-K.

Name(last,first,middle): Smith, Larry, D
Title & Dept:            assistant professor;  architecture
Work Addr & Phone:       MIT;  3-6000
Home Addr & Phone:       1234 West End Dr., Arlington, MA;  648-9000
Electronic Mail Addr:

NEW: WORK ADDR & PHONE:    NE43-500, MIT;  3-6000
     Electronic Mail Addr: LDS at MIT-XX

     submitted:  6-06-83          by:  Smith, Larry, D
     marked:  seen                by:  Davis, Alice          on:  6-10-83
     marked:  valid               by:  Davis, Alice          on:  6-15-83

Comments: telephoned at 3-6000 on 6-15-83; ok


COMMANDS: ?,↑E,↑N,↑P,<DEL>,↑W,↑K,↑R,↑Q,↑T,ES-S,ES-U,ES-V,ES-I,ES-W
```

Figure 3-11: Sample Update Edit Window

---

status windows remain on the screen and have the same functions as when the update survey window is on the screen. The echo area is now blank. As in the DIRSYS update request window, the typed characters will go into the window rather than into the echo area. The update edit window contains a brief explanation of the editing commands, which are the same as those that are used when a request is submitted using DIRSYS. CTRL-N or a carriage return and CTRL-P move the highlight to next and previous lines, respectively, <DEL> deletes a character, CTRL-W deletes a word, CTRL-K "kills" a line of information, and CTRL-E exits the update edit window. The marking commands are also available in the update edit window.

3.1.3 Features that Illustrate Design Principles

This section describes, from the designer's point of view, features of the DIRSYS interface that illustrate the design principles that were presented in Section 3.1.1.

1. **User's Model**: The interface was designed to simulate the scanning of a paper phone book -- a process familiar to most DIRSYS users. Like a paper phone book, the electronic phone book is organized as "pages" of information, with information for each person occupying one line on a page. Users can scan up or down a page and forward or back through the pages as with a paper phone book. Thus, users should be able to form a clear conceptual model of the directory system, and this model should aid them in learning to use the system.

   In addition, the interface shares certain features with full-screen display editors such as Emacs [51]. Emacs uses, for example, a status line and an echo area, a similar command structure, similar cursor movement and editing commands, and an incremental search. Thus, experienced computer users who are familiar with full-screen editors should be able to form a clear model of the system by drawing on their knowledge of those editors.

2. **Minimal Command Set**: The DIRSYS command set is small, but it contains all the commands necessary for using the system. In addition, the commands are not redundant. The user, for example, does not have to choose between different but equivalent ways to get help, quit the program, or redisplay the current screen. In several instances, however, the result of executing several commands may be the same as executing one alternate command. Using the <DEL> command several times could have the same effect as using the CTRL-W command; several CTRL-W commands could have the same effect as one CTRL-K command. In these cases, the CTRL-W and CTRL-K commands were included for convenience and did not seem to increase the command set to an unmanageable size. (This assertion, however, needs to be tested.)

3. **Simple Command Structure**: The DIRSYS command structure is simple. Each command name is derived from an English phrase that describes the command function and is abbreviated using the first letter of the phrase (except the help command, which is issued by typing "?"). For example, the Next Line command is abbreviated with an "N". Hence, commands are easy to remember and easy to type. A command is issued by holding down the <CTRL> key while at the same time typing the command abbreviation. To exit the current screen, for example, the user types CTRL-E, and to start the tutorial, the user types CTRL-T. The

<CTRL> key is used in order to distinguish commands from text. In this way, commands still can be abbreviated by single letters, and alphabetic characters can be consistently interpreted as text. The only commands that do not require using the <CTRL> key are the help command ("?") and the request modification command, ESC-M, which is issued by hitting the <ESC> key then typing the letter M. The "?" is used for help because it is short, suggestive, and easy to remember. The <ESC> key is used instead of the <CTRL> key because typing CTRL-M is equivalent on many terminals to hitting the carriage return or enter key. It was judged undesirable to start the modification process with a carriage return because inexperienced users sometimes hit a carriage return after typing a name, and having the system display an update request window as that point would be disconcerting.

4. **Consistent Command Interpretation**: Commands in DIRSYS do not have different meanings in different contexts; they always perform the same actions. The CTRL-R command, for example, always redisplays the current screen, and the CTRL-Q command always quits the program, while "?" always causes a help window to appear. Similarly, alphabetic characters typed without using the <CTRL> or <ESC> keys always are interpreted as text. If commands are not available in all contexts, they have the same function in the contexts in which they are available. The CTRL-W command, for example, is not available in the herald, in the tutorial, or in the second level help window. Where it is available, however, it always has the same function, that of deleting a word.

5. **Help Facility**: The user can get help at any time by typing "?" or CTRL-T. Both commands allow the user to get help easily without interrupting the immediate task. The help window that appears by typing "?" contains information specific to the user's current context; it contains a list of currently available commands. Similarly, the tutorial, started by typing CTRL-T, begins at the section most relevant to the user's current context. If the user types CTRL-T from the update window, for example, the tutorial starts with the section describing how to request an update. Finally, different levels of help are provided. Users who want a detailed description of the system can read the tutorial, while those who want only a very brief explanation of the commands can use the first level of the help facility. Those who want more explanation, but not as much as in the tutorial, can use the second level of the help facility. The second help level, easily reached by typing a second "?" and one of the commands listed in the first help window, gives a detailed explanation of a specific command.

6. **Feedback**: The system provides several types of feedback to the user. Characters that the user types are displayed in the echo area, and matching entries are highlighted by displaying them in reverse video, in uppercase letters, and with periods between the fields of information. The system provides the user with prompts, such as the one displayed in the echo area and the one displayed after the user has typed a "?" to display a second level of help. In addition, DIRSYS sends messages to the user via the status window. The messages are concise, polite, understandable, and informative. They are displayed in a conspicuous place (on the top line of the status window) and do not interfere with the other windows on the screen. They always appear in the same place in order to take advantage of the user's tendency to associate different meanings with different areas on the screen [4, 14]. Messages are sent to indicate which command DIRSYS is working on if the command is not immediately executed. When the CTRL-F command is issued, for example, a message is displayed which says, "Getting next screen, please wait...". (This sort of message is not necessary with the CTRL-N command because the highlight is moved immediately to the next line.) When the matching entry is not on the screen, DIRSYS sends the message, "Searching for *x*, please wait...", where *x* is what the user has typed. Should the user type an unavailable command, DIRSYS responds by ringing the terminal's bell and sending a message saying that *x* is not available, where *x* is the command that was typed, and suggesting that the user type "?" for help. This message remains on the top line of the status window until the user types something or for four seconds, whichever comes first. The bottom line of the status window contains a list of available commands.

   The user also can provide feedback to the system manager by sending messages to an address that is listed in the herald, in the tutorial, and in the message that is displayed when the user stops the program.[40]

7. **Error Handling**: When the system was designed, a wide variety of user errors were anticipated. Consequently, it is very difficult to cause DIRSYS to terminate abnormally. If an error occurs that causes the system to fail, the error is written to a file, and a message is sent to the maintainers. The user is not given details about the problem, but is told, "Sorry, DIRSYS isn't working correctly. The maintainers have been notified, and the problem should be fixed soon. Thank you for using DIRSYS." The system is then stopped. Thus, the underlying aspects of the system remain hidden from the user.

---

[40] Currently, the address is an electronic mailing address.

3.2 The Database

This section is divided into two parts. The first one describes the structure of the database and the indexes. The second one describes how the database is searched using the indexes.

3.2.1 Database and Index Structures

As mentioned earlier, it was desired that the database *access method*, i.e., the database structure and database search mechanism[41], be simple and fast enough to keep up with user requests. Moreover, it was desirable that the database *access time*, i.e., the time to search the database and retrieve the desired record, be relatively constant for all types of user requests. To meet this latter requirement, the access method needed to facilitate both sequential and random accessing of records, since the records are accessed sequentially when a next or previous screen command is given, and a record is accessed randomly when the user types a name. Then after a single record is accessed randomly, successive records are accessed sequentially in order to fill the screen.

An *indexed sequential* access method meets the above requirements [52]. It consists of a sequential file of records, ordered alphabetically by name, and several levels of indexes (in this case, three).

The first level of indexes, called the *alphabetic pointers*, logically divides the database into *alphabetic groups*, groups of records that contain last names beginning with the same letter. Thus, there are 26 entries, one for each letter of the alphabet.

The second level of indexes, called the *name index*, logically partitions the alphabetic groups into equally sized groups of records, called *record blocks*. (The optimal block size is determined through performance testing and is discussed in the next chapter.)

---

[41] The search mechanism is the algorithm for accessing records in the database.

The beginning record in each block is used to form a name index entry by pairing its record number with a substring of the name that it contains. The substring, which must be unique, is formed by taking beginning characters from the name, which is written as "last name,first name,middle name". A substring may be any length; it need contain only enough characters to make it unique. Testing showed that for the database being used, the substrings would usually be about four characters long.

The third level of indexes, called the *record pointers*, contains an entry for every record in the database.[42] It maps record numbers into physical addresses in the database. The records are of variable length; therefore, a method for determining the location of each record was needed.

The next section discusses how these indexes are used to locate a record in the database.

3.2.2 Searching the Database

A record in the database is accessed by locating the appropriate alphabetic pointer -- the one associated with the beginning letter of the sought after last name -- and following the pointer to the name index. The name index is traversed sequentially from that starting point until an index value, i.e., a name substring, is found that is greater than or equal to the search name. The record associated with that index entry is located in the database by using the record number in that entry and the record pointers. The database then is searched sequentially from that starting point until the desired record is located. The number of records that have to be searched is always less than or equal to the number of records in a record block.

An example will illustrate this indexing scheme. Assume that the database and index are as pictured in Figure 3-12 and that the user has typed "Babb" into the echo area. The entry in the alphabetic pointers corresponding to "B" is located and is used as

---

[42]There are approximately 18,000 records in the database.

an offset into the name index. In Figure 3-12, the name index entry at this offset contains the string "Baa" and the number 102. Successive name index strings are compared with the typed string until an index string is found that is greater than or equal to the typed string. To be "equal" means that the typed string is identical to the index string or that it begins with the same characters that are in the index string. An example in the next paragraph further explains this case. In this example, the search of the name index ends at the entry containing the index string "Babe" and the number 112. The typed string is now guaranteed to be between record number 107 and record number 112 since "Babb" is alphabetically between "Baba" and "Babe". Record number 107 is chosen as the beginning point for a forward search of the database. The first matching record will be positioned at the top margin of the screen as discussed in the Section 3.1.2.

If the typed string is equal to an index string, i.e., is identical to an index string or begins with the same characters that make up the index string, the method for determining where to start searching in the database is a little different. If the user had typed "Babee", for example, the search of the index again would have ended on the index entry containing the string "Babe" and the number 112. This time, the record could be between either 107 and 112 or between 112 or 117. It would be between 107 and 112, for example, if record 112 contained the name "Babel". It would be between 112 and 117, for example, if record 112 contained the name, "Babea". Therefore, the typed string must be compared to the full name in the database that is associated with the index string. If the typed string is greater than the full name, record 107 is chosen as the beginning point for a search of the database; otherwise, record 112 is chosen.

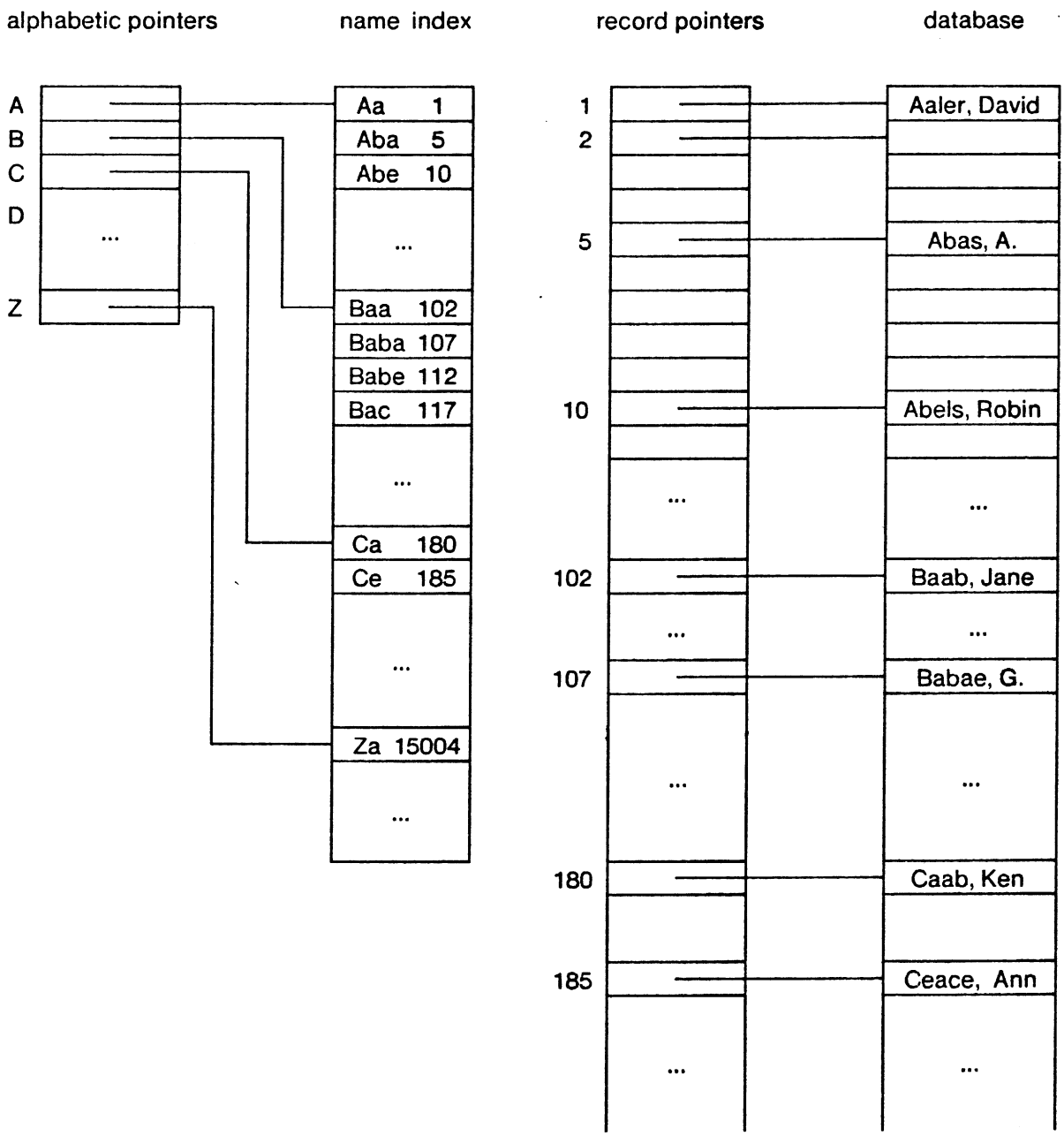


Figure 3-12: Sample Database and Indexes

3.3 The Update File

As discussed earlier, users may submit update requests to the DIRSYS manager. The manager then validates or invalidates the requests. The *update daemon*, or *daemon*, is the program that runs at night to add the information in the requests to the database. This section, divided into two parts, describes what happens to the update requests from the time the user submits the requests until the requests are added to the database. The first part of the section describes the update file structure. The second part of the section describes the addition of the update requests to the database.

3.3.1 Update File Structure

The update file is separate from the database. In this way, nothing is ever written into the database that DIRSYS is using. Each update request that is submitted becomes a record in the update file. The update record contains: the record number, the current database record, the new information, the name of the person submitting the update request[43], the date that the request was submitted, a flag indicating whether the request has been marked seen or unseen, the date that the request was marked seen or unseen and the name of the person who did the marking, a flag indicating the status of the request, i.e., valid, invalid, or waiting, the date when the status was marked and the name of the person who did the marking. (See Figure 3-13.) When a request is added to the update file, it is automatically marked as unseen and waiting for verification. When the manager views the request in the update survey window or update edit window (discussed in Section 3.1), he can mark the requests as seen or unseen and valid, invalid, or waiting for verification.

---

[43] It is assumed that an authentication server can get this information.

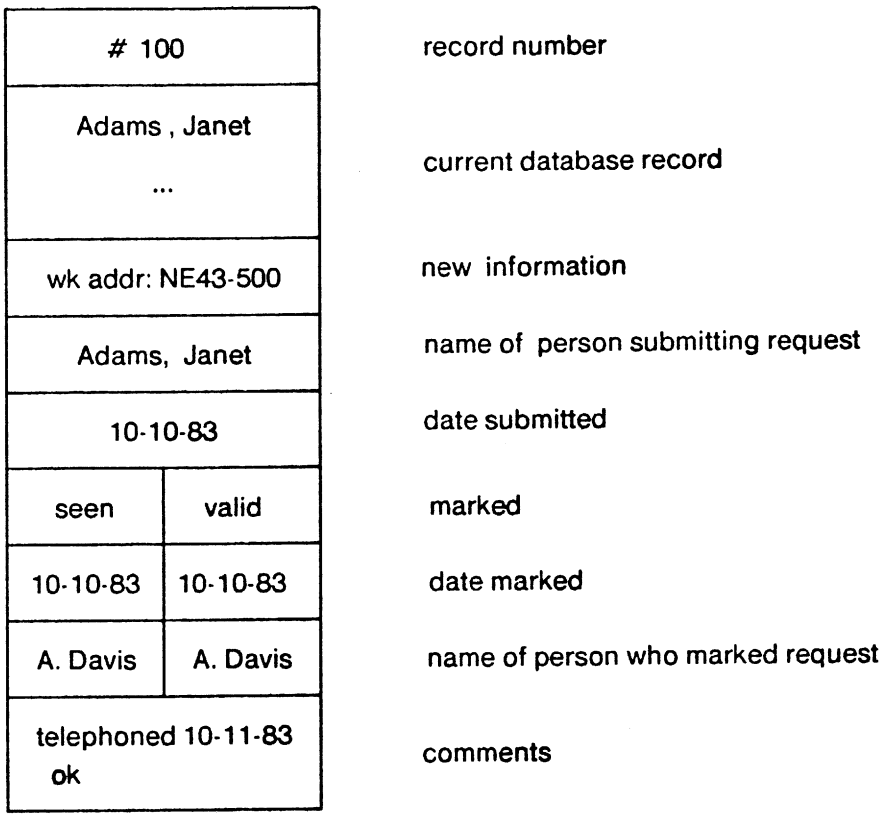


Figure 3-13: Sample Update Record

3.3.2 Adding Updates to the Database

The update daemon, which runs at night, removes the invalid requests from the update file, rebuilds the database and indexes, incorporating the valid update requests, and builds a new update file with the requests marked waiting. It also logs the changes that it has made and sends the manager a message detailing the transactions, if any, that took place.

The actual daemon procedure is as follows. If there are any unseen requests, the daemon notifies the manager via electronic mail. It then treats unseen requests as waiting requests. In fact, most of the unseen requests will be marked waiting; an

unseen request that is marked valid or invalid will be the exception.[44] The daemon then removes the invalid requests from the update file, copies them into the log, and alphabetizes the remaining requests. Should a request contain new information for the name field, then that request is alphabetized using the new name instead of the old one. In this way, the new records will be in the correct alphabetical order in the new database. (An example presented later in this section explains this case further.)

The daemon then checks for conflicting update requests, i.e., requests that contain different information for the same field of a database record.[45] It marks any conflicting update requests waiting, logs the conflict, and notifies the manager. The daemon also checks for update requests that could be combined into one request. For example, it consolidates update requests that contain new information for different fields of a database record.

The daemon now is ready to rebuild the database and indexes. It reads a record from the database and searches the update file for a valid update request that contains a record that should precede the record that was just read.[46] If such a request is found, the new information in the request is combined with the database record contained in the update request, and this new record is copied into the new database.[47]

Next, the daemon checks to see if the record that was just read from the database should be copied into the new database by searching the update file for valid

---

[44] If an unseen request is marked valid or invalid, the manager either forgot to mark it seen after the validation process or left it marked unseen in order to be reminded of it, but forgot to mark it waiting. In either case, the daemon waits for the requests to be correctly marked before acting on them.

[45] The record numbers kept in the update requests facilitate locating requests for the same database record.

[46] The requests were alphabetized to facilitate this process.

[47] Before copying new information into the database record contained in the update record, the daemon makes sure that that database record is identical to the corresponding record in the current database.

requests that contain that record number. If no matching record numbers are found, then the database record apparently still contains up-to-date information and is copied into the new database. If a valid request does contain a matching record number, then the database record is not copied into the new database because it will be replaced by a new record. If the request containing the matching record number is marked waiting, then the record number in the update request is modified if necessary, and the database record is copied into the update request and into the new database.[48] The waiting update request then is copied into a new update file.

The daemon keeps a table that maps old record numbers into new record numbers so that records that were identified in the old database by record number can be located in the new database. This table is used to modify record numbers in waiting update requests and new update requests that are submitted while the update procedure is in progress.

The daemon continues until the above procedure has been carried out for the last record in the database. Then it repeats the process once more to take care of any new records that should appear after the last record in the database. Finally, it sends a message to the manager saying that the new database and indexes have been built.

The next morning, the manager reviews the messages from the daemon, tests the new database to make sure DIRSYS can use it, and switches DIRSYS to the new database. The update list associated with the old database is checked to see if any update requests were added while the database was being rebuilt. If so, the new update requests are added to the newly created update list, after modifying the record numbers contained in them to reflect any changes in the order of the records in the new database.[49] DIRSYS is then told to use the new update file. Again, care

---

[48] If both valid and waiting requests exist for the same record number, then the new database record containing the modifications in the valid update request is copied into the waiting update request.

[49] Records may be in a different order in the new database if records have been added or deleted or if names in any of the records have been spelled differently.

must be taken to check for new update requests before switching over to the new update file.

A sample update file and the pertinent part of the database before the daemon has run are shown in Figure 3-14. The following example illustrates how a new database and update file would be built. The new update file and pertinent part of the database after the daemon has run are shown in Figure 3-15.

The update requests shown in Figure 3-14 have been alphabetized. Note that the third update request involves changing the spelling of a name in such a way that the record should appear in a new alphabetical location in the database. The request, therefore, was alphabetized using the new name, "Adems, John", instead of the old one, "Adams, John".

The daemon starts the update procedure by reading a record from the database -- record 100, "Adams, Janet". It does not find a record in the update file that should precede that record, but does find a valid request to modify that record. The "Adams, Janet" record, therefore, is not copied into the new database since a new record for "Adams, Janet" will be copied into the new database in the next step. The daemon next reads record 101, "Adams, John", and locates a valid update request containing a record that should precede "Adams, John" in the database. This update request contains the "Adams, Janet" record. The new "Adams, Janet" record, formed by combining the new information with the database record contained in the request, is copied into the new database. The daemon does not copy the "Adams, John" record into the new database because there is a valid request to modify that record. The daemon reads the next record in the database -- record 102, "Adelson, David". There is no request that contains a record that should precede "Adelson, David" since the change in spelling from "Adams" to "Adems" causes the new "Adems, John" record to succeed the "Adelson, David" record. The daemon then locates a request to modify the "Adelson, David" record, but it is marked waiting, so the daemon copies the current record into the new

database, changes the record number in the waiting request from 102 to 101 to reflect the new record order, and copies the waiting request into the new update file. The daemon then reads record 103, "Ades, Katherine", locates the update request for a record that should precede that record, adds that record -- the "Adems, John" record -- to the database, then adds the record for "Ades, Katherine". The process continues until the daemon has read all the records in the database and all the requests in the update file and has copied the appropriate records into the new database while at the same time rebuilding the indexes.

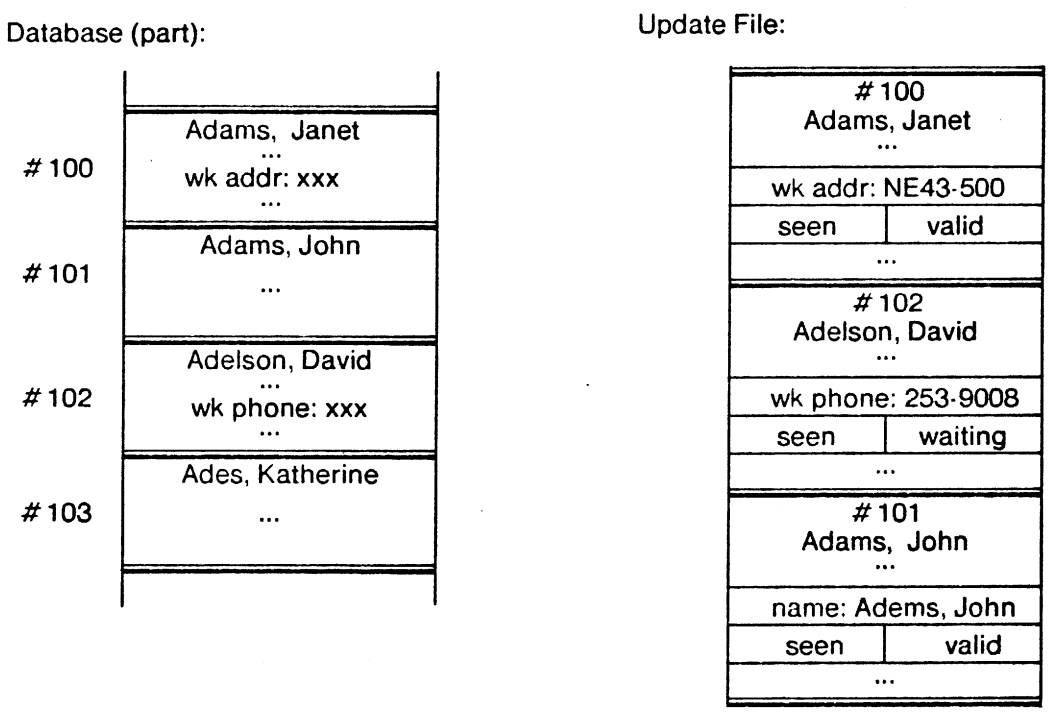


Figure 3-14: Sample Update File and Database Before Daemon Runs

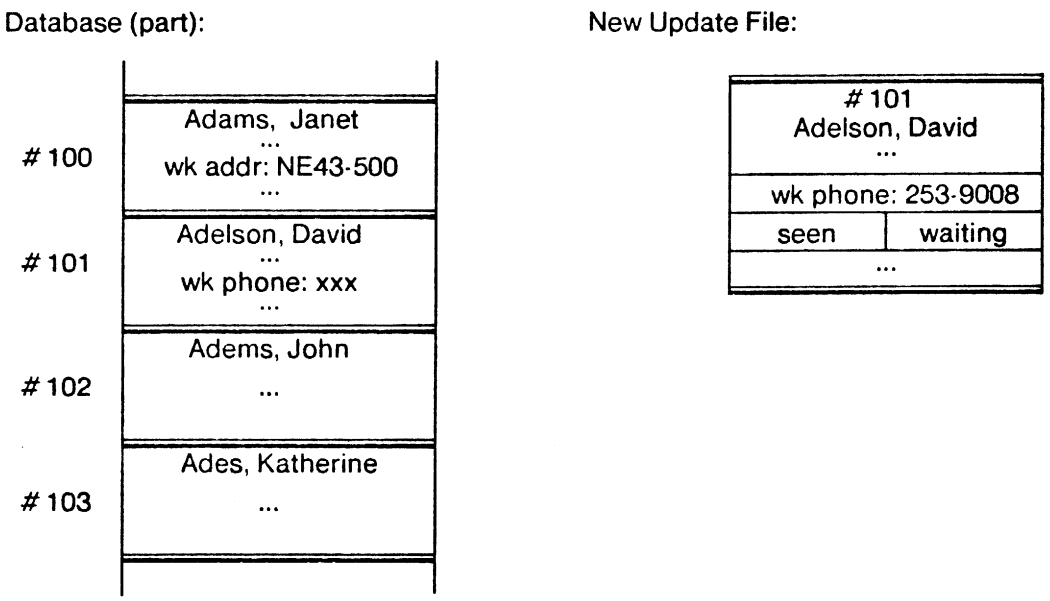


Figure 3-15: New Update File and Database After Daemon Runs

Chapter Four

Preliminary Evaluation

A preliminary evaluation of the interface and the database access method was made. The evaluation included observing people use the system and taking timing measurements on the database access method. The update daemon has not been evaluated yet.

4.1 User Interface

The interface was evaluated informally by observing people use it, recording their comments while using it, and asking them questions about it. Ten people were observed, three of whom had had little or no computer experience. The other seven were experienced computer users.

Starting DIRSYS

There were mixed reactions as to the usefulness of the herald screen. All the naive computer users read it and said that they liked having an introductory screen that told them how to get started and that listed useful commands. Three of the experienced computer users read the herald; the remaining four preferred to start the program immediately and to find out what commands did by trying them. One experienced computer user suggested that it would be convenient to avoid the herald screen if someone already knew how to use the system. One could type, for example, "DIRSYS Smith, Jane", and the system would display the appropriate page of the phone book with the closest matching record highlighted.[50]

---

[50]This option will be considered for the next version of the system.

Using the Commands

All users thought that the commands were easy to type. They seemed to have no trouble with the <CTRL> or <ESC> keys, or with typing characters to search for a name. The users were able to learn quickly how to use the commands. The naive computer users thought the commands easy to remember because they are mnemonic. Some of the experienced computer users found the commands difficult to remember because the DIRSYS commands are similar to Emacs commands but not identical. In Emacs, for example, users type CTRL-L to redisplay a screen, rather than CTRL-R, and type CTRL-F to move forward a character instead of forward a screen. They type CTRL-V to move forward a screen in Emacs.[51] Thus, in the choice of command abbreviation there is a tradeoff between the needs of naive computer users, who often would like mnemonic commands, and the needs of experienced computer users, who might prefer commands that are identical to commands in other systems.

Only two users, both of whom are experienced computer users, used the CTRL-W command. One other experienced computer user said that he thought the CTRL-W command would be useful for an experienced DIRSYS user, but that first-time users did not really need it since they could use the <DEL> key or the CTRL-K command. The users seemed to like having the CTRL-K command for clearing the echo area because it was convenient, even though, as one user pointed out, it functioned in the same way as repeatedly using the <DEL> key.

Several users wanted more commands. One user suggested a command to specify a department name as well as a personal name when searching for a record. Consideration was given to including this feature in DIRSYS, but adding the option was postponed in order to keep the input of search information simple. (Searching

---

[51] Perhaps DIRSYS could read an initialization file from the user's computer directory as Emacs does and provide a default initialization for users who do not start DIRSYS from a system on which they have a directory. In this way, experienced users could rename the DIRSYS commands if they desired.

the directory using information besides personal names is discussed in Section 5.2.) Two other users, both of whom are Emacs users, wanted a command that would move the highlight to every fourth record. Such a command would be similar to the Emacs CTRL-U command which causes the command that follows it to be repeated four times.

Finally, the users were able to anticipate what a command would do in a different context, and they did not seem to be bothered by not having all commands available in all contexts.

Looking Up a Name

All the users expressed a liking for the search screen layout. They thought that the label window was necessary both to demarcate the main window and to give useful information such as column headings. They reported that the echo area located at the bottom of the screen close to the keyboard was helpful for keeping track of what they had typed.

The users expressed a liking for an area set aside for system messages (the top line of the status window) and for having it located near the echo area where they could pay attention to information in both areas at the same time. They regarded the system messages as informative and especially liked having a list of available commands displayed. They appreciated being told that the system was still working on a command if the command was not executed immediately and being told when they had typed an unavailable command. One user suggested that error messages remain on the screen until something was typed. Most other users thought that error messages should disappear from the screen after the messages had been read. The current scheme is a compromise -- error messages disappear as soon as the user types something or after four seconds, whichever comes first. The messages were more easily noticed when they were reverse videoed and accompanied by a ringing of the terminal bell. Most users liked the terminal bell because it let them know that the system was sending them a message. Two users found the bell annoying. Some

studies suggest that systems should avoid using the terminal bell because the users might be embarrassed, thinking that the bell called attention to their mistakes [47].

All users were able to look up entries within minutes of starting DIRSYS. They indicated that they liked searching for a name by simply typing characters and liked the idea of highlighting the matching record. The reverse video was seen as an improvement over highlighting without it. They also liked being able to move the highlight manually. One suggestion dealing with the highlight was to use the uppercase letters for highlighting only if the terminal did not have reverse video capabilities. In this way, the names would be capitalized as people are used to seeing them, with both uppercase and lowercase letters.

When DIRSYS is retrieving the new screenful of records, there is a delay of about three seconds.[52] This delay may be due to the sequential design of the program or to the program's spending an excessive amount of time reading and processing records.[53] Most users did not find the delay objectionable because the system sent a message saying that it was searching for the desired record. It must be noted, however, that the naive computer users did not know what kind of response time to expect. In addition, the experienced computer users might have found the response time more objectionable if they had not been accustomed to using interactive systems on time-sharing machines.

The users found the delay objectionable, however, when DIRSYS displayed an intermediate screen before it displayed the screen containing the desired record. If the characters were not typed quickly, the system started searching for the record that matched the first few characters that the user had typed (as an incremental interface should). There was then a delay while the system retrieved and displayed

---

[52]The three second delay is for a terminal with 24 lines. The delay would be greater for a terminal with a longer screen since more records would have to be retrieved to fill the screen.

[53]Further testing for the cause of this delay is outside the scope of this thesis since system response time is good enough for a preliminary evaluation of the system.

the intermediate screen. The characters that the user had typed while DIRSYS was retrieving and displaying the records could not be echoed until the new screen of records had been displayed. People found this delay annoying because they could not keep track of what characters had been typed. This problem became especially bothersome when someone made a typing error and had to wait for the characters to appear before the mistake could be corrected.

The display of an intermediate screen was especially annoying when the user moved the highlight and then typed characters. For example, if the highlight were placed on "Johnson, D", and the user then typed "Johnson, J", depending on how fast the characters were typed, the system might only get "Johns" before it started displaying a new screen of entries. A previous page containing the first record matching "Johns" would be displayed before the page containing the record matching "Johnson, J". Most users found this display of a previous page followed by the display of the correct page disconcerting. The same display behavior occurred when deleting characters. If the highlight were on, "Johnson, D", and the user deleted "D", and then typed "J", DIRSYS might display the screen containing the first record matching "Johnson," before displaying the screen containing the record matching "Johnson, J".

Most users liked the fact that deleting the last character from the echo area did not cause the highlight to move. All of the naive computer users expected the highlight to remain where it was. Most of the experienced computer users expected the highlight to go back to the previously highlighted entry because of the way that the Emacs incremental search facility works, but most also said that they would rather have the highlight remain where it was. One thought it would go back to the beginning of the database.

Because of the display delays, it is difficult to say whether or not the incremental interface is a good interface for this directory system. If the redisplay were faster or could be interrupted as in Emacs so that the echoing of characters could be

immediate, then the intermediate screen display might not be annoying.[54] Alternately, the system could wait until the user had finished typing characters before displaying the appropriate page of the directory in order to decrease the flashing of the screen. The interface then, however, would not appear to be incremental. All of the naive computer users expressed a preference for a system that would search only when they told it to. One said that he was not in control of the system, that the computer was in control when it searched for the matching record as he typed characters. (Perhaps if the system response time had been better then he might have felt more in control.) The experienced computer users are more familiar with incremental searching, since many full-screen editors use it, and most of them liked the idea of using it to search the directory.

Getting Help

All three naive computer users read the tutorial and thought it was well worded. One thought that it was too long. Another pointed out that he felt as if he had to remember all the terms defined in the tutorial, e.g., "echo area", to be able to use the system. This point is a good one; the tutorial could be rewritten without using specific terms. None of the experienced computer users read the tutorial, but they all thought that it was helpful to have it for those who wanted to read about the system before experimenting with commands.

All users typed the help command ("?") at least twice, and, as expected, the novices used the help facility more often than the experienced computer users. The users thought that the help facility was useful, well worded, and easy to understand. They were able to understand that a command typed from the first level help window would be executed in the previous window. There were mixed reactions, however, to what occurred after typing CTRL-T or ESC-M from the help window. When one of these commands was typed, the directory data window was redisplayed before the tutorial

---

[54] In Emacs, when the user types a character or command, any screen display that is in progress is interrupted while the user's input is processed.

or update request windows were displayed in an attempt to let the user know that he would return to the directory data window after exiting the tutorial or update facility. Some users thought that the redisplay of the directory data window caused unnecessary delay and screen flashing. Others did not mind the redisplay, but were not sure that it helped them figure out what window would be redisplayed when they exited the tutorial or update request windows. An interface without redisplay of the directory data window should be implemented so that users can compare the two designs.

The users liked having two levels of help and did not seem to mind having to return to the first help level before being able to execute commands in the window that was present before entering the help facility. The users found the command prompt for the second level help window that appears in the echo area helpful but thought that it was difficult to notice unless it was reverse videoed. One user said that a more detailed explanation level would be helpful. He had overlooked the explanation of the second level help because the explanation was not in the same format as the other commands; it was written as a statement near the bottom of the help window.[55]

Requesting an Update

Most of the users submitted an update request and thought that the update facility was easy to use.[56] They thought it convenient to be able to send changes directly to the DIRSYS manager. They understood that they were submitting requests for modification rather than modifying the directory database and understood that the requests would be verified by the manager.

---

[55]The command will be explained differently in a new version of the interface.

[56]The information in the database is not up-to-date so most users submitted new information for their directory entries.

4.2 Database Access Method

Preliminary timings were taken to discover how record block size affects the database access time and to determine an optimal record block size.[57] A random sample of 200 names was chosen from the database. Each name was located in the database using the indexing scheme discussed in Section 3.2, and the following quantities were counted: the number of disk accesses while searching the index, the number of disk accesses while searching the database, the central processor time while searching the index, and the central processor time while searching the database.[58] The disk accesses counted, however, might not have been actual physical disk accesses because the counting process did not take into account the possibility that the operating system might have had the desired information cached in memory. The timed searches were repeated on two other files of sample names that had been constructed by randomly reordering the names in the first sample file. The timed quantities for the three files were then averaged.

The above procedure was carried out using indexes that had been constructed for record block sizes ranging from 5 to 100 in increments of 5. The searches using the 20 indexes were then repeated for internal buffer sizes ranging from 128 to 2048 words.

It was predicted that 30 would be a suitable record block size based on statistics that indicated that block sizes larger than 25 resulted in a significant decrease in the average number of characters needed for unique index entries. A relatively small index was judged to be desirable in order to minimize the number of disk accesses required to search the index. In addition, calculations with disk access time and average record size indicated that reading a maximum of 30 records before locating the desired entry would keep the response time acceptable.

---

[57] As mentioned in Section 3.2, the name index logically partitions the groups of records containing last names with the same first letter into equally sized groups of records, called record blocks.

[58] DIRSYS is implemented on a DEC-SYSTEM 20 running the TOPS-20 and on a VAX 11/750 running UNIX. The timings were taken on the DEC-SYSTEM 20 because it had a higher resolution timer than the VAX 11/750.

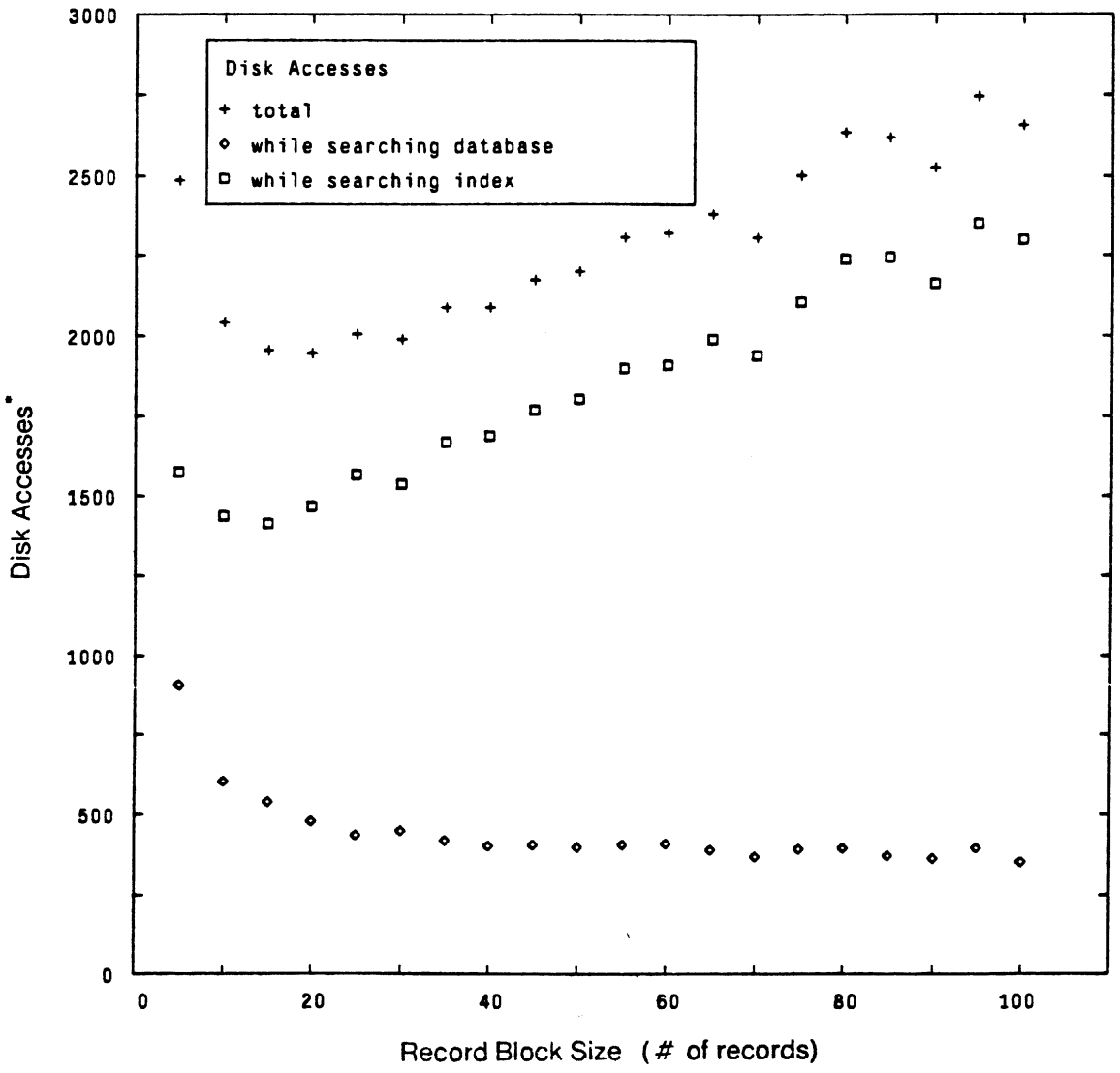
Results of the timing process are shown in Figures 4-1 and 4-2 and in Appendix B. The graphs of disk accesses versus record block size, one of which is shown in Figure 4-1, seem to indicate that for record block sizes larger than about 20, the number of disk accesses required to read the index is relatively constant at approximately two per search. The increase in disk accesses below about 20 is due to a large index. It takes more disk accesses to read the index, and with more index entries, the probably of having an index entry that matches the search name increases, in which case, a record is read from the database to determine whether the desired record is before or after that particular record in the database.[59] Most disk accesses for record block sizes larger than about 20 are due to reading records from the database, as expected. Also as expected, the larger the buffer size, the fewer average number of disk accesses required to read the database since more information is kept in memory. However, this decrease in disk accesses is only beneficial when doing many searches in a row or when moving to the next or previous page in the directory. If the user only wants to look up one name, reading in a lot of extra information, and causing the user to wait, is unnecessary.

The graphs of central processor time versus record block size, one of which is shown in Figure 4-2, indicate that the system spends very little time searching the index. Most of its time is spent reading and processing the records of information. The increase in central processor time below a record block size of about 20 is due to having to search a large index. This search time probably could be decreased if the index were not searched sequentially. As expected, the central processor time seems to be independent of internal buffer size.

The results of the database timings indicate that for record block sizes above 20 the index is small enough to be searched quickly. Additional testing is needed to determine an optimal internal buffer size.

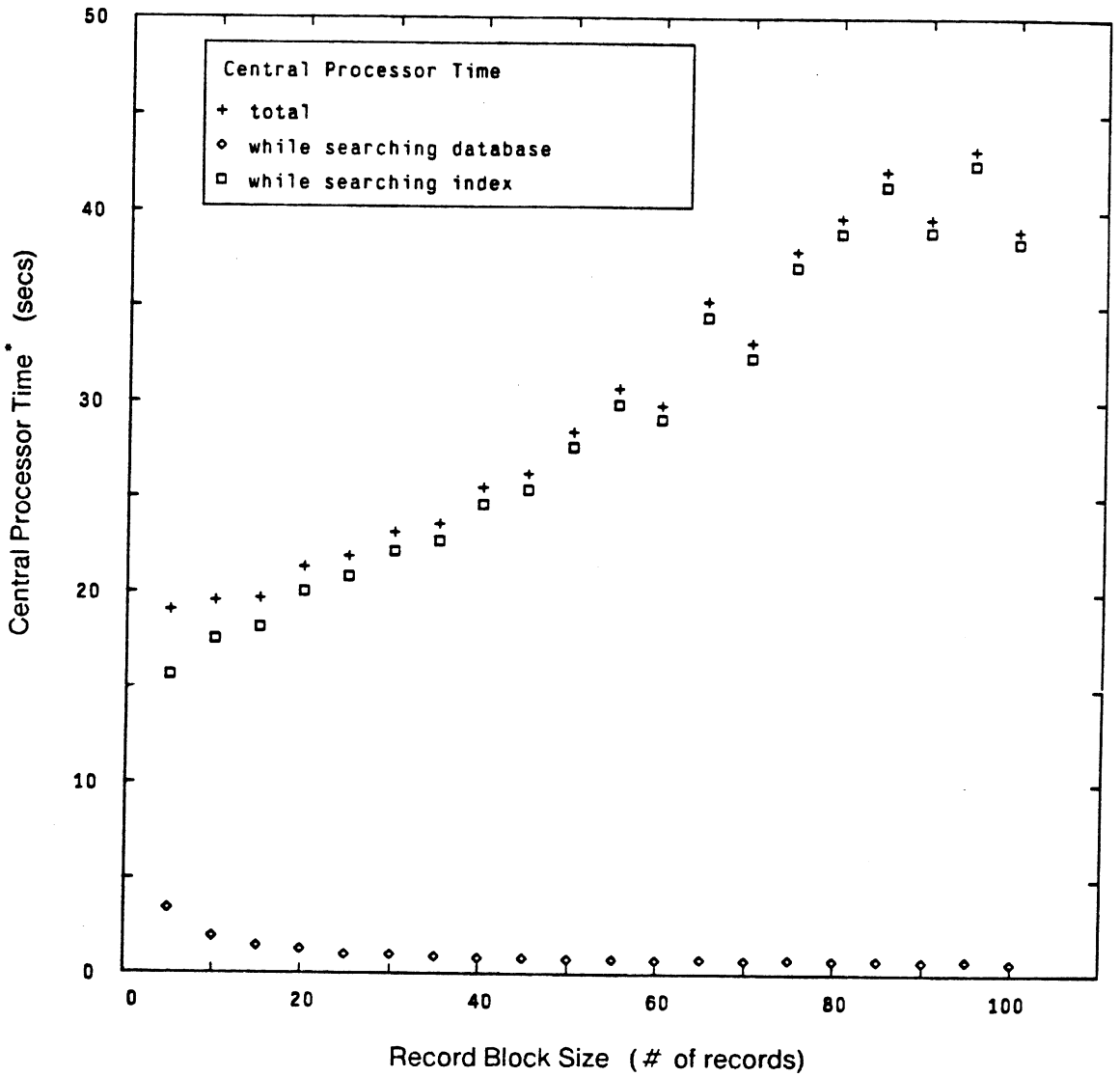
---

[59]The search algorithm is discussed in Section 3.2.



* for 200 names

Figure 4-1: Disk Accesses vs. Record Block Size, Buffer Size 128



* for 200 names

Figure 4-2: Central Processor Time vs. Record Block Size, Buffer Size 128

Chapter Five

Conclusions and Future Work

5.1 Conclusions

This thesis has described the design and implementation of an online directory assistance system called DIRSYS. The system, designed for use by members of the M.I.T. community, has an *incremental* interface that combines features of a paper phone book with those of a full-screen editor such as Emacs [51]. Each directory entry is displayed in a compact one line per entry format, as are entries in a paper phone book. Since more information is available for each entry than in a paper phone book, a command is available for changing entries into a more expanded format in which additional information is displayed.

Users may "browse" through this electronic phone book by issuing commands similar to Emacs' cursor motion commands, or they may search for a specific name by typing the name. After each letter that the user types, DIRSYS moves the *highlight*, the means for emphasizing an entry, to the entry whose name string most closely matches what the user has typed so far. This incremental search mechanism is similar to that used in Emacs. The system provides a *help facility* with two levels: the first level reminds users of which commands are available; the second level describes the function of a specified command in detail. A *tutorial* is available for users who want a very detailed description of the system.

Finally, DIRSYS provides a facility for keeping the information in the directory database up-to-date. A user may submit *update requests*, which contain information about modifications to his directory entry, to the DIRSYS manager. When the update requests have been validated, the information contained in them is incorporated into the directory database by an *update daemon*, a program that runs every night to update the database.

A preliminary evaluation of DIRSYS indicates that the system can be used easily by both inexperienced and experienced computer users. Details of the evaluation were presented in Chapter 4. The learning aids guide the novice without encumbering the experienced user. The commands are simple, easy to use, and consistently interpreted. The system provides prompts and polite, informative messages to the user. It also is robust in that it is very difficult to cause DIRSYS to fail to operate. Finally, individuals who used DIRSYS seemed to enjoy using the system, even though the system response was slow at times, and agreed that a directory system such as DIRSYS would provide a convenient service for use both inside and outside the M.I.T. community.

5.2 Future Work

This section is divided into two parts. The first describes work that could be done on the current system; the second describes possible extensions to the current system.

5.2.1 Work on the Current System

Quantitative Evaluation

DIRSYS was designed to be easy to learn to use and easy to use once it had been learned. The preliminary evaluation indicates that these goals were met; still a more thorough quantitative evaluation is needed. Moreover, the evaluation should include experienced DIRSYS users.

Ease of learning is often measured by determining the average time required for people to learn to use a system. An alternative would be to measure the percentage of people who could learn to use a system in a given period of time. In testing for ease of learning, it is important to define what is meant by "to learn to use the system". The phrase could mean being able to do basic tasks, but not necessarily quickly; it could mean being able to perform basic tasks quickly; it could mean being able to do basic tasks quickly and some advanced tasks; it could mean being able to

perform both basic and advanced tasks quickly. It is also necessary to identify the basic and advanced tasks. With DIRSYS, for example, submitting an update might be considered an advanced task.

Ease of use can be measured by determining the speed with which a person can use a system, e.g., by finding the average time required to complete a task or the percentage of a task completed within a given time period. Alternatively, one could measure ease of use by counting the number of errors that a person makes while using the system. There is evidence to suggest that the fewer the number of errors, the easier the system is to use [28]. With DIRSYS, however, it would be difficult to determine when users made errors, especially since users often may experiment with commands. In testing for ease of use it is important to distinguish between inexperienced and experienced computer users, because knowledge of computers is being tested as well as knowledge of the system. Collecting statistics about user sessions, e.g., keeping track of everything a user types, would also help determine which commands people find most useful. Unfortunately, only statistics for experienced computer users can be collected now since inexperienced computer users currently do not have easy access to the system.

In addition to measuring ease of learning and ease of use, it would be interesting to measure user attitudes toward DIRSYS. There is an abundance of literature on attitude measurement [11, 22, 29, 41, 54]. One method that has been successfully used for evaluating attitudes toward computer systems is the semantic differential technique [15, 18, 31, 50]. This technique measures people's reactions to various concepts in terms of ratings on bipolar, seven-step scales defined at their extremes by contrasting adjectives. A typical semantic differential looks like:

easy: - - - - : - - - - : - - - - : - - - - : - - - - : - - - - : - - - - : difficult.

The seven positions on the scale correspond to (from left to right) extremely easy, quite easy, slightly easy, equally easy and difficult, slightly difficult, quite difficult, and extremely difficult. The subject is asked to evaluate a concept, e.g., using DIRSYS,

by placing a check mark on one of the positions on the scale. The collected data can then be analyzed mathematically by assigning values of 1 to 7 or 3 to -3 to the scale.

In addition, the database access method needs to be evaluated further. It would be useful to find out how the operating system on which DIRSYS runs affects system performance. The operating system's paging scheme may prevent requests for new information from being actual physical disk accesses because the operating system may have the desired information cached in memory. It would also be interesting to see how much of a performance difference alternate database organizations would make.

Finally, the update procedure has not been tested yet. The update daemon takes about three hours to rebuild the database and indexes on a dedicated VAX 11/750 running the UNIX operating system. Statistics need to be taken to determine how the rebuild procedure affects response time if DIRSYS is being used at the same time that the database is being rebuilt. It also would be useful to know how often update requests were received in order to determine whether or not updating the database once every 24 hours is appropriate. It would be interesting to compare the current update procedure with one that did not rebuild the entire database. One possible design would be to organize the database as several files and rebuild only the files that were affected by the updates. This scheme would be more efficient than rebuilding the entire database if the percentage of files that have to be rebuilt is small. Having the database organized as multiple files, however, might introduce delays in reading and displaying the records for the incremental interface.

Improved Performance

The performance of DIRSYS needs to be improved by eliminating the delay when the system retrieves and displays records. The cause of the delay needs to be identified by determining exactly where in the program the time is being spent. As noted in Chapter 4, the delay may be due to the sequential design of the program or to the program's spending an excessive amount of time reading and processing records.

One solution might be to allow display to be interrupted when another command or character is waiting to be processed. Another possible cause of delay may be that DIRSYS relies on screen handling routines in the implementation language library rather than updating the screen itself.[60] Thus, further investigation of system performance is needed.

Improved Methods for Building and Maintaining the Database

In order for DIRSYS to be a usable service, a convenient way to set up the database is needed. Currently it is difficult to transform the information from the registrar's and personnel offices into a directory database. The process involves reading and merging the information from magnetic tapes, editing it to remove format inconsistencies, merging it with electronic mail address information, and converting it into a more compact format. An automatic way of getting the directory information in a convenient format is needed. Ideally, the registrar's and personnel offices would keep directory information in similar formats, including electronic mail address information, and would send it over a data line that connected those offices to the directory system. The directory system then could update its directory database with the new information. With this plan, it also would be possible to send copies of the validated update requests for DIRSYS to the registrar's and personnel offices in order for their records to be kept up-to-date as well.

5.2.2 Extensions to the Current System

Running DIRSYS on Multiple Machines

DIRSYS is designed to run on a single dedicated computer to which users connect in order to run the system. Alternately, DIRSYS could run on more than one computer. The availability of the service would be increased with such a plan since if something went wrong with one machine, the other machine(s) could continue to operate the

---

[60] DIRSYS is implemented in CLU [30].

system. In addition, the system load could be distributed. When a user types "DIRSYS" to start the system, a program on the user's machine could find the DIRSYS machine with the least number of users, set up a connection to that machine, and start DIRSYS on it. By distributing the load, one machine could do the updating of the database while the other machine(s) continued to provide the directory service. The machine doing the updating would be assigned fewer users than the others in order to keep system response fast on that machine. If one machine were to be taken offline to update the database, then a different update scheme could be employed since not writing directly into the database would no longer be a requirement. (In the current scheme, as discussed in Chapter 2, in order not to "disturb" the database that DIRSYS is using, a new database is built by copying the old database and merging update information into it.)

Finally, if DIRSYS were running on more than one machine, the update requests would have to be propagated to the update files kept on each machine. In addition, the update program would have to make sure that there were no inconsistencies in the update files when it began to rebuild the database. Care also would need to be taken when copying the new database and new update file to the machines. Any update requests that had been submitted while the database was being updated would need to be added to the new update file.[61]

The above scheme assumes that all the processing, e.g., screen handling and database searching, is done on the dedicated machines. An alternate plan is to distribute the processing involved in running DIRSYS between central machines and user sites. For example, copies of the database could be kept on several central machines, and the screen handling and input processing routines could be kept on personal computers connected to the machines via a network.[62] The centralized

---

[61]The record numbers in the new update requests might have modified, as discussed in Section 3.3.2.

[62]This plan assumes that the network would have a high enough bandwidth to handle the data transfer.

machines would not need to be dedicated to running only DIRSYS since all processing would be done elsewhere. Expanding the system to be able to serve more users would not require adding another centralized machine, only another copy of the database. The system could be accessed from personal computers distributed throughout the M.I.T. community. In addition, users could obtain a copy of the DIRSYS program and run the system on their own machines.

More Sophisticated Matching Schemes

There are several sophisticated name matching schemes that could be added to the system in order to facilitate locating a desired record. These alternate matching schemes become especially important if the desired record is not found using the incremental search. These features could be provided as separate commands or as one command that automatically searches using all of the alternate matching schemes. The alternate matching schemes could include the following. The system could match first names and nicknames by searching a table of common nicknames. It could search for records containing names that matched the typed first, middle, and last names and initials in any order. For example, the record for someone who goes by "Ann Smith" might be listed in the directory database under the name "Karen Ann Smith". In this case, if DIRSYS were to search the database looking for "Ann" as the middle name instead of the first name, the desired record would be located. The system could provide a *phonetic search* facility that would locate records containing names that "sound like" the typed name. Names could be encoded in such a way that similar sounding names have the same code. One such encoding method is the Soundex method, developed by Margaret K. Odel and Robert C. Russell.[63]

One possible way in which the alternate matching schemes could be used is the following. If after typing a name, a matching record is not found by means of

---

[63] *U.S. Patents 1261167* (1918), *1435663* (1922). Cited by [26].

incremental search, the user could issue a PHONETIC SEARCH command which would cause records to be displayed containing names that phonetically match the typed name. The information would be displayed in compressed format as with the incremental search screen, and all commands available with the incremental search screen would be available with the phonetic search screen. The phonetic search facility could be "turned off" automatically when a new search is started or by issuing a specific command. Thus, adding alternate matching schemes to DIRSYS would not necessarily require drastic changes in the interface design.

Other matching schemes that could be added include matching records using fields of information besides personal name. If someone is looking for the directory entry for John Smith who is in the Department of Psychology, for example, being able to input the department information as well as the personal name might shorten the list of matching entries. A simple way to input the department information would be needed. In one such scheme, a list of numerical department abbreviations could be displayed on a section of the screen, and the user could type the department number instead of the department name. This scheme shortens typing time and gets rid of the need for sophisticated methods of interpreting user input. The user could type the department number, then start typing a name. There would be no confusion about what part of the user's input was department information since personal names do not contain numbers. Indexing the database by department as well as by personal names might facilitate locating and displaying matching records quickly.

A Mouse Interface

Many current systems have interfaces that employ a *mouse*, a small mechanical device which, when moved around on a flat surface such as a desk top, moves a pointer on a terminal screen. A mouse often has one or more buttons on its top or side for selecting options, e.g., for issuing commands. One such interface has been designed for DIRSYS and is summarized briefly here [24].

With this DIRSYS interface, the pointer on the screen that moves when the mouse is

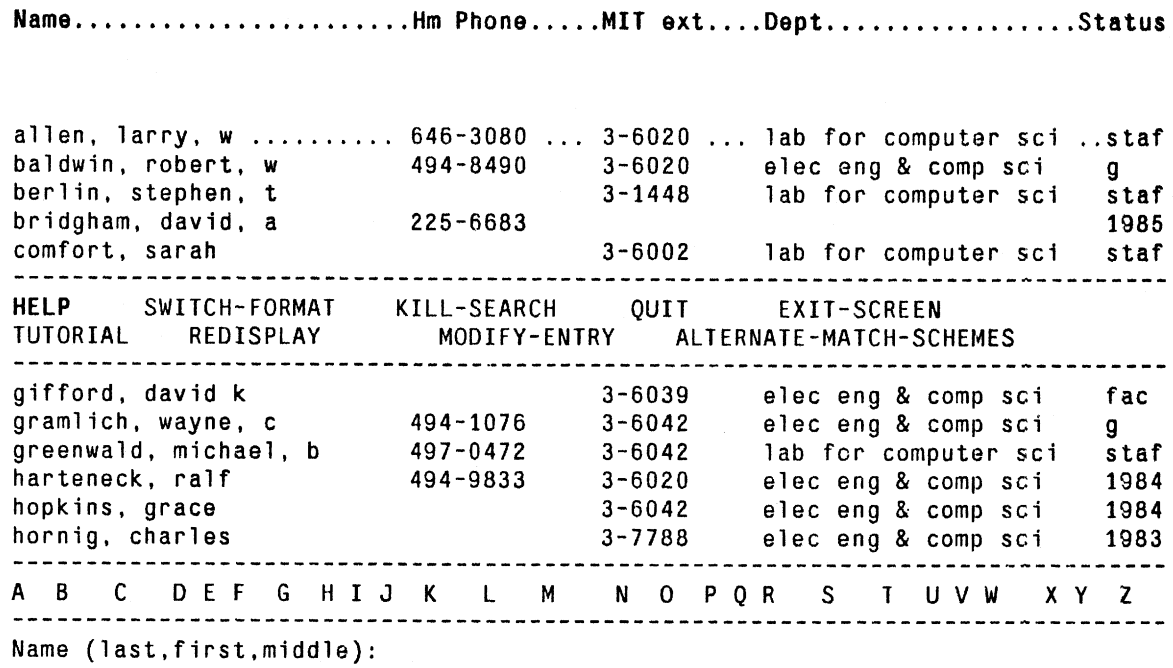


Figure 5-1: Sample Search Screen for Mouse Interface

moved is the highlight. To highlight an entry on the screen, the mouse is moved until the highlight rests on that entry. To display the next or previous page of entries, the mouse is moved so that the highlight reaches the bottom or top, respectively, of the directory data window. "Clicking" one of the buttons on the mouse causes a pop-up window to appear wherever the highlight is located. The pop-up window contains a list of available commands. (See Figure 5-1). To issue one of the commands, the highlight is moved to the desired command and a button on the mouse is clicked. Only the command name would be highlighted, not the entire line containing the command. Depending on how many buttons the mouse has, frequently used commands, such as EXIT SCREEN and SWITCH FORMAT, could be executed by clicking different buttons on the mouse. To begin an incremental search, the user could either type the name into the echo area or move the highlight to a position on a bar graph representing the distribution of alphabetic groups in the database. The bar graph could be displayed, for example, on the right-hand-side or at the bottom of the screen, as shown in Figure 5-1. The graph would be labeled as a dictionary is

labeled, with letters proportionally spaced out along the graph to indicate where the last names beginning with those letters are located in the directory. In order to locate the record for John Smith, for example, the user could place the mouse pointer half way between the "S" and the "T" on the graph and click the mouse button. A little trial and error might be needed to find the exact location of the Smith records.

An Interface for Outside M.I.T.

A non-incremental interface for use outside the M.I.T. community can be designed in such a way as to enforce the Institutes' privacy policy. (As mentioned in Chapter 2, the privacy policy states that the phone book is not to be made publicly available, especially for commercial purposes.) The incremental DIRSYS interface can be modified to restrict the number of directory entries that users outside the M.I.T. community may view. One method for achieving this restriction would be for DIRSYS to display entries only if the non-M.I.T. user has completed typing the name and hit an <ENTER> key and only if the number of matching entries is less than a certain number. That number would be the maximum number of records that someone outside the M.I.T. community would be allowed to view and would be set by individuals at M.I.T. who determine the privacy policy. If there are more matching records than can be displayed, the system could ask the user to input more information, or it could ask the user a specific question about information that would distinguish the records. It might ask, for example, if the person whose name the user had typed is a student or a member of the faculty or staff, or it might list the department names contained in the matching entries and ask the user with which department the person is associated.

Implementing an interface for use outside M.I.T. would require a reliable means of distinguishing M.I.T. users from non-M.I.T. users. There are currently several ways to make this distinction, each of which exhibits a tradeoff between security and convenience. The system could keep a list of authorized M.I.T. users and require those users to input a personal password when starting DIRSYS. Alternately, the

system could operate on the assumption that anyone who has access to an M.I.T. computer is an authorized M.I.T. user.[64] This second scheme would be more convenient for the users, but would not allow as much control over who used the M.I.T. interface to the system.

A Mailer Interface

Another interface that could be added to DIRSYS is an interface for a mailer. This interface would simplify the sending of electronic mail by allowing someone to send a message addressed using only a personal name and the name of the Institute. One could send a message, for example, addressed to "Jane Anderson at MIT", and the mailer would look up Jane Anderson in the directory database, find her electronic mail address, and forward the message to that address. In order to provide this mail forwarding service, the mailer must be able to identify a unique recipient name before sending mail. If a user were to send mail to a person at M.I.T. using a name that did not match exactly a name in the M.I.T. database, the system could behave in one of several ways. It could refuse to send the mail and instead send the user a message which would include information about the closest possible match (or matches) and a request to use the exact name listed in the entry (or one of the entries) to resend the mail. If the name could be inexactly matched with one of the entries in the database, e.g., using a nickname, the system could send the mail to the associated electronic mail address and send the user a message giving the person's name as listed in the database entry, requesting that future mail be sent using that name. A person not listed in the database would still be able to receive mail, but the sender would have to know the recipient's electronic mail address rather than relying on the forwarding service to respond to the person's name.

---

[64] An M.I.T. computer can be identified by its network address.

Yellow Pages

A useful addition to the directory assistance service would be a "yellow pages" section for Institute information. To provide a good "yellow pages" service requires more than looking for an entry that matches exactly what the user has typed. A sophisticated indexing scheme is needed that would enable information to be located when asked for by a name different from the one given in the directory. For example, the system should be able to locate an entry for the Health Center even though in the directory listing it is called the Medical Department. Also needed is a way for users to "browse" through the information since they may not know what information they are looking for, but might recognize it if they saw it.

Except for its slow performance, DIRSYS is usable and robust. The only other effort needed to put the system into service is to improve the method for building and maintaining the database. While the present design and implementation are complete, the features suggested here may be valuable additions for improvement.

Appendix A

Glossary of Terms

The following is a glossary of terms that are used in this thesis.

Access method: The database structure, i.e., the physical storage structure, and the search mechanism, i.e., the algorithm for accessing records in the database.

Access time: The time to search a database and retrieve the desired record.

Alphabetic groups: Groups of database records that contain last names beginning with the same letter.

Alphabetic pointers: The first of DIRSYS' three levels indexes. The alphabetic pointers logically divide the database into *alphabetic groups*.

Directory data window: In DIRSYS, the window in the center of the screen, between the label and status windows, that contains directory information in compressed or expanded format. It is also called the data window.

DIRSYS: The name of the directory system. (The name is derived from DIRectory SYStem.)

DIRSYS manager: The person in charge of maintaining DIRSYS, including validating and marking update requests.

Echo area: In DIRSYS, this is the bottom-most window and the last line on the physical screen. When the user types characters, they are shown in this window.

Help window: In DIRSYS, the temporary window that appears when the user types "?". The first level help window contains an explanation of the available commands;

the second level help window contains a detailed explanation of a specified command.

Herald: The first screen displayed when DIRSYS is started. It contains a brief description of what the system does and how to start using it and a list of commands needed by the first-time user, along with the functions of those commands. It also gives an address to which users may send comments.

Highlight: The means by which DIRSYS emphasizes an entry on the screen. A highlighted entry is displayed in reverse video if the terminal has that capability and in uppercase letters with periods between the fields of information.

Incremental search: When a system begins searching for a matching string after each character typed by the user.

Indexed sequential access method: A database access method that consists of a sequential file of ordered records and an ordered index, or several levels of ordered indexes, for accessing those records.

Label window: In DIRSYS, this is the top-most line on the physical screen. It is shown in reverse video if the terminal has that capability and generally contains information about the window directly below it.

Mark an update: The operation done by the DIRSYS manager to indicate whether an update request is seen or unseen and valid, invalid, or waiting for verification.

Name index: The second of DIRSYS' three levels of indexes. It logically partitions the alphabetic groups in the database into equally sized groups of records, called record blocks. The beginning record in each block is used to form a name index entry by pairing its record number with a substring of the name that it contains.

Non-incremental search: When a system begins searching for a matching string after the user has typed the complete string.

Pop-up window: A window that appears when the user issues a command, e.g., a help command, and disappears when the user begins typing again.

Record blocks: Equally sized groups of database records.

Record pointers: The third of DIRSYS' three levels of indexes. The record pointers contain an entry for every record in the database and map record numbers into physical addresses.

Status window: In DIRSYS, this window is located directly above the echo area and consists of two lines, both of which are shown in reverse video if the terminal has that capability. The upper line is used for sending messages to the user; the lower line is used for listing available commands.

Survey updates: The operation available to the DIRSYS manager for viewing the update requests that have been submitted.

Tutorial window: In DIRSYS, the window that replaces the directory data window when CTRL-T is typed. It contains the text of the tutorial.

Update daemon: The program that runs at night to add information in the update requests to the DIRSYS database.

Update edit window: The window that is used by the manager to edit and mark an update request.

Update request window: In DIRSYS, the window that replaces the data directory window when ESC-M is typed and contains a directory entry to be modified. An update request is submitted by typing CTRL-U.

Update request: A message that a user sends to the DIRSYS manager requesting that his entry in the DIRSYS database be modified.

Update status: The status of an update request, i.e., whether it is valid, invalid, or waiting for verification.

Update survey window: The window that is used by the manager to view and mark update requests in compressed or expanded format.

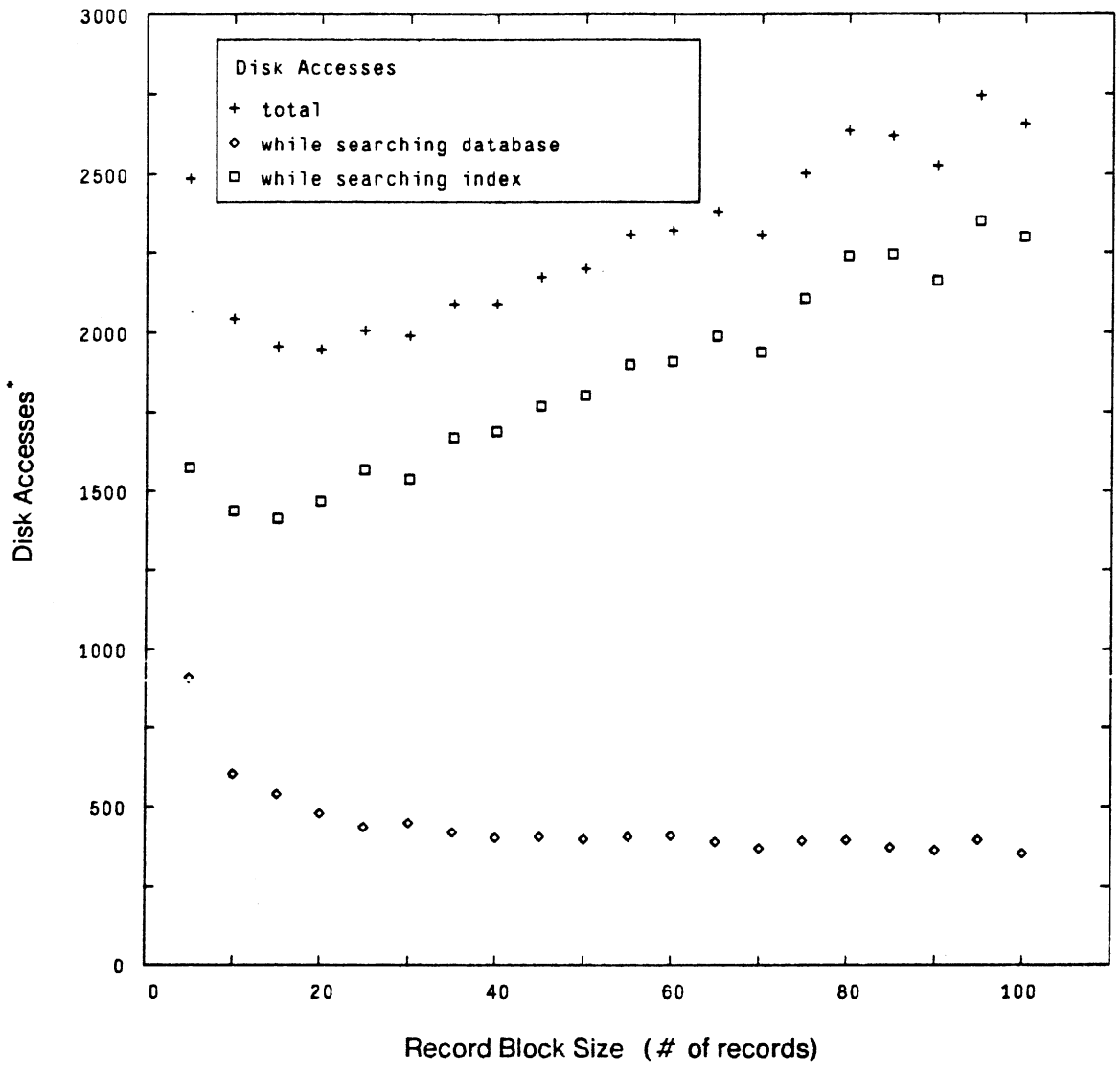
Window: A rectangular area on a physical video screen.

Appendix B

Performance Graphs

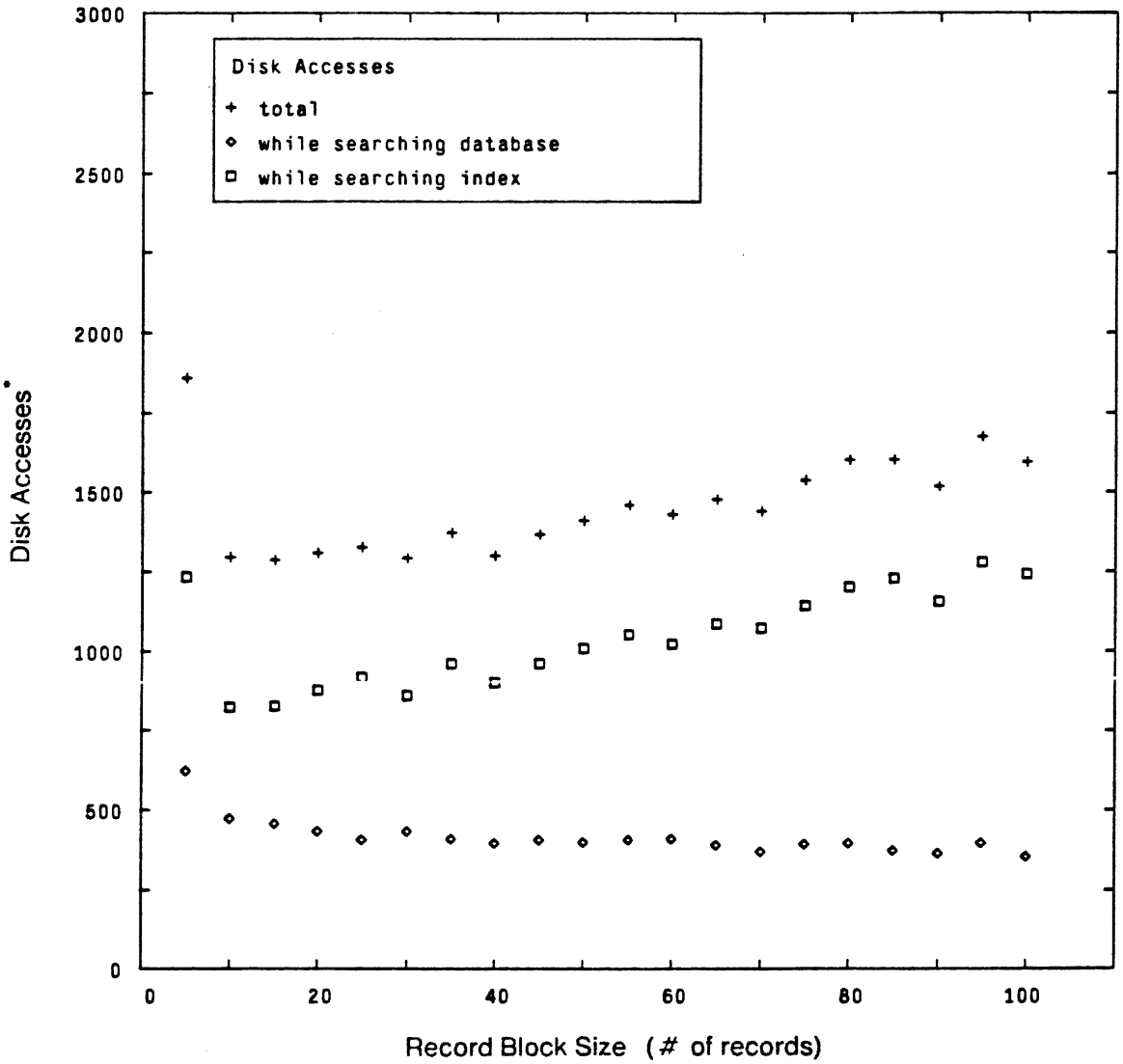
The following graphs, discussed in Section 4.2, are the results of preliminary timings that were taken to discover how record block size affects the database access time and to determine an optimal block size. A random sample of 200 names was chosen from the database. Each name was located in the database using the indexing scheme discussed in Section 3.2, and the following quantities were counted for several internal buffer sizes: the number of disk accesses while searching the index, the number of disk accesses while searching the database, the central processor time while searching the index, and the central processor time while searching the database. (The disk accesses counted, however, might not have been actual physical disk accesses because the counting process did not take into account the possibility that the operating system might have had the desired information cached in memory.)

The graphs indicate that for record block sizes above 20 the index is small enough to be searched quickly. Additional testing is needed to determine an optimal internal buffer size. (See Section 4.2 for a more detailed discussion of the timings.)



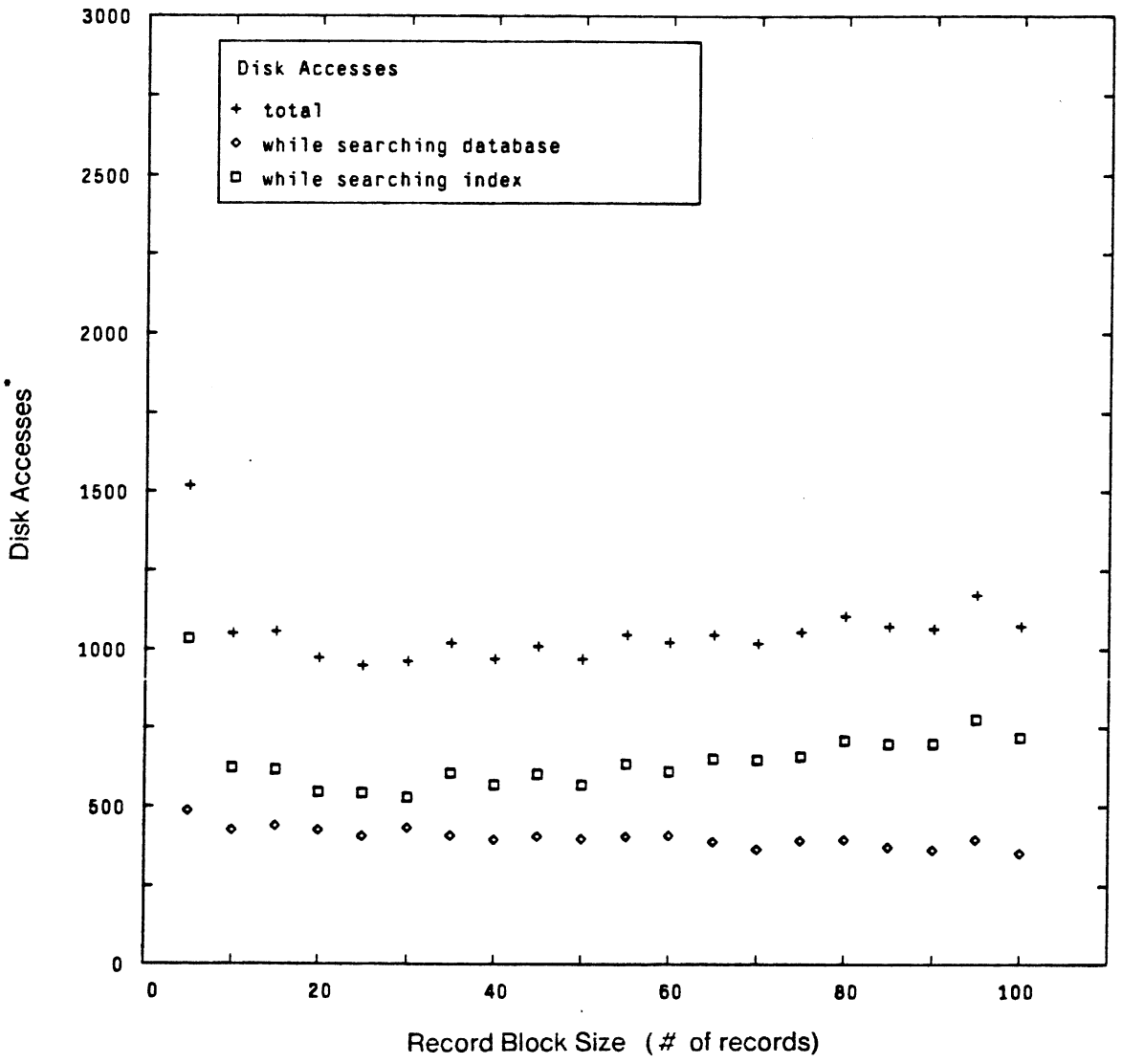
* for 200 names

Figure B-1: Disk Accesses vs. Record Block Size, Buffer Size 128



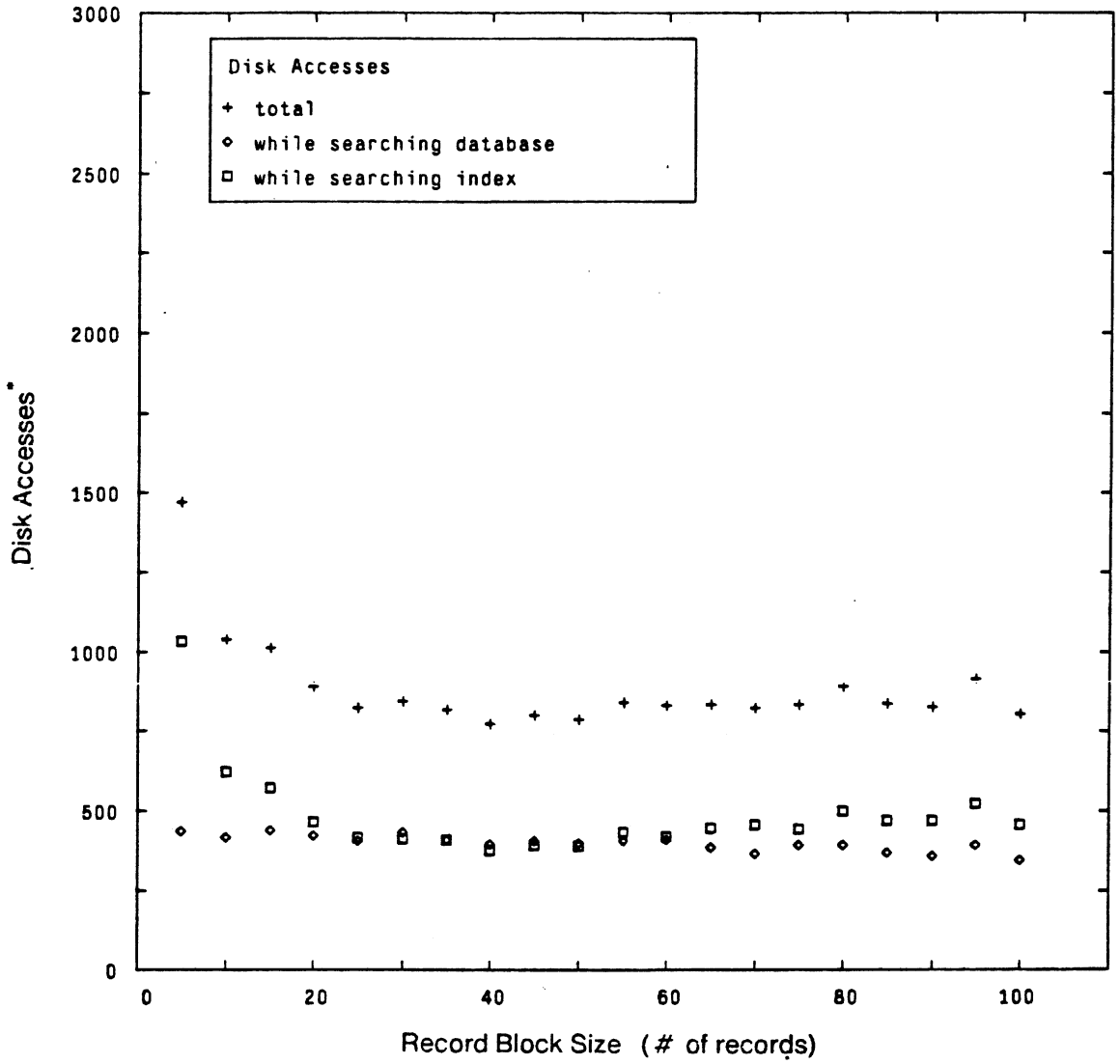
* for 200 names

Figure B-2: Disk Accesses vs. Record Block Size, Buffer Size 256



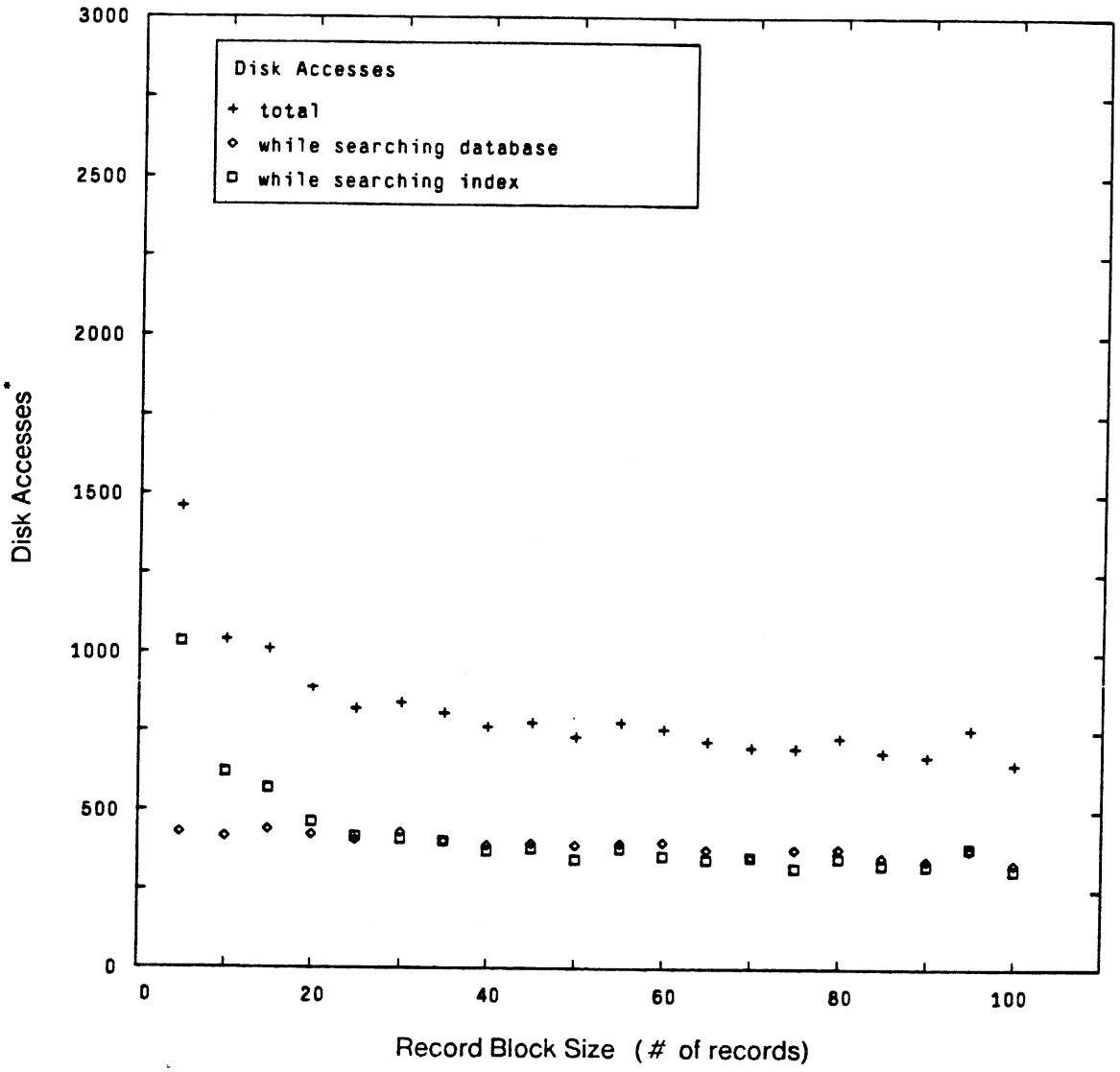
* for 200 names

Figure B-3: Disk Accesses vs. Record Block Size, Buffer Size 512



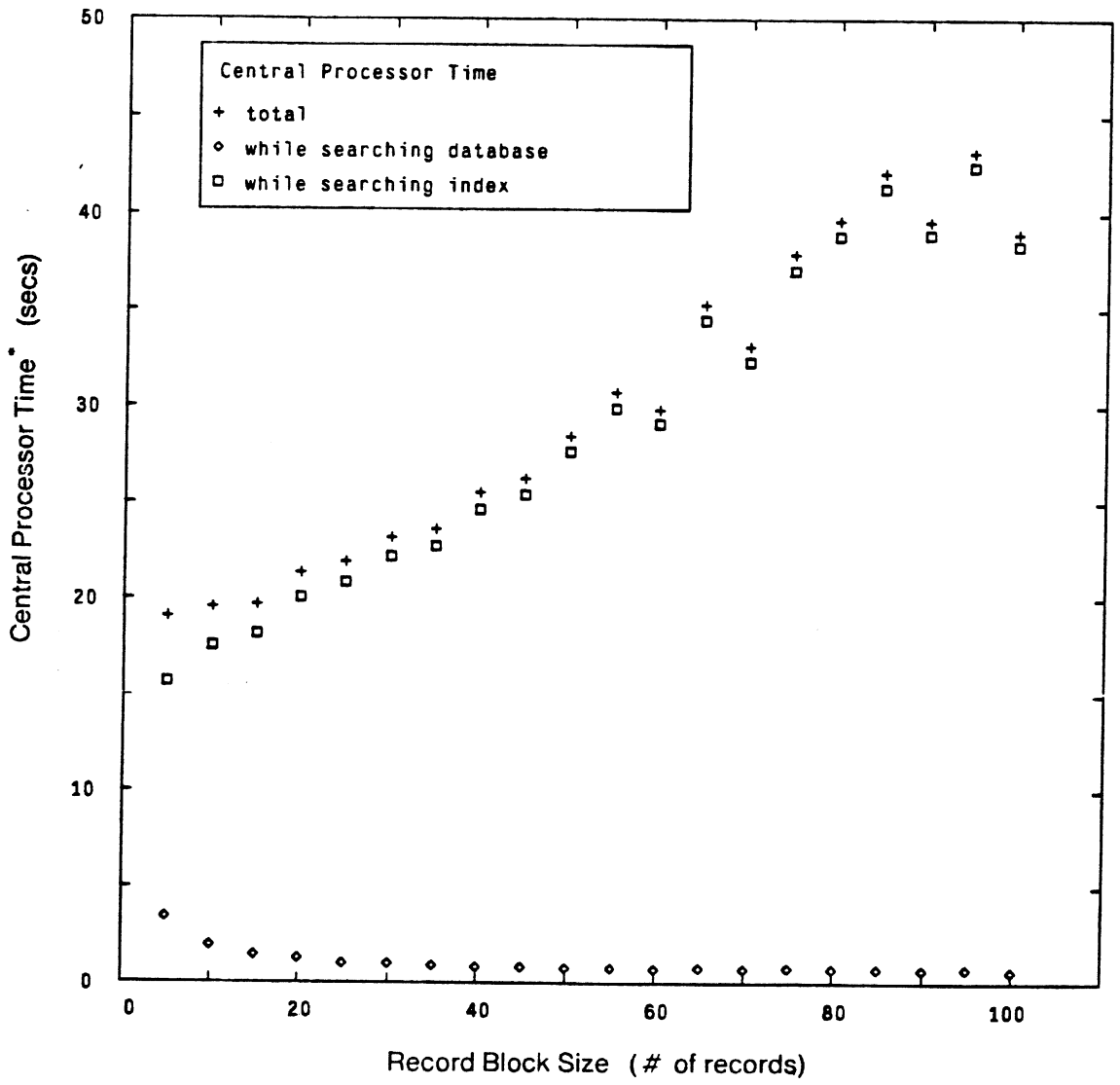
* for 200 names

Figure B-4: Disk Accesses vs. Record Block Size, Buffer Size 1024



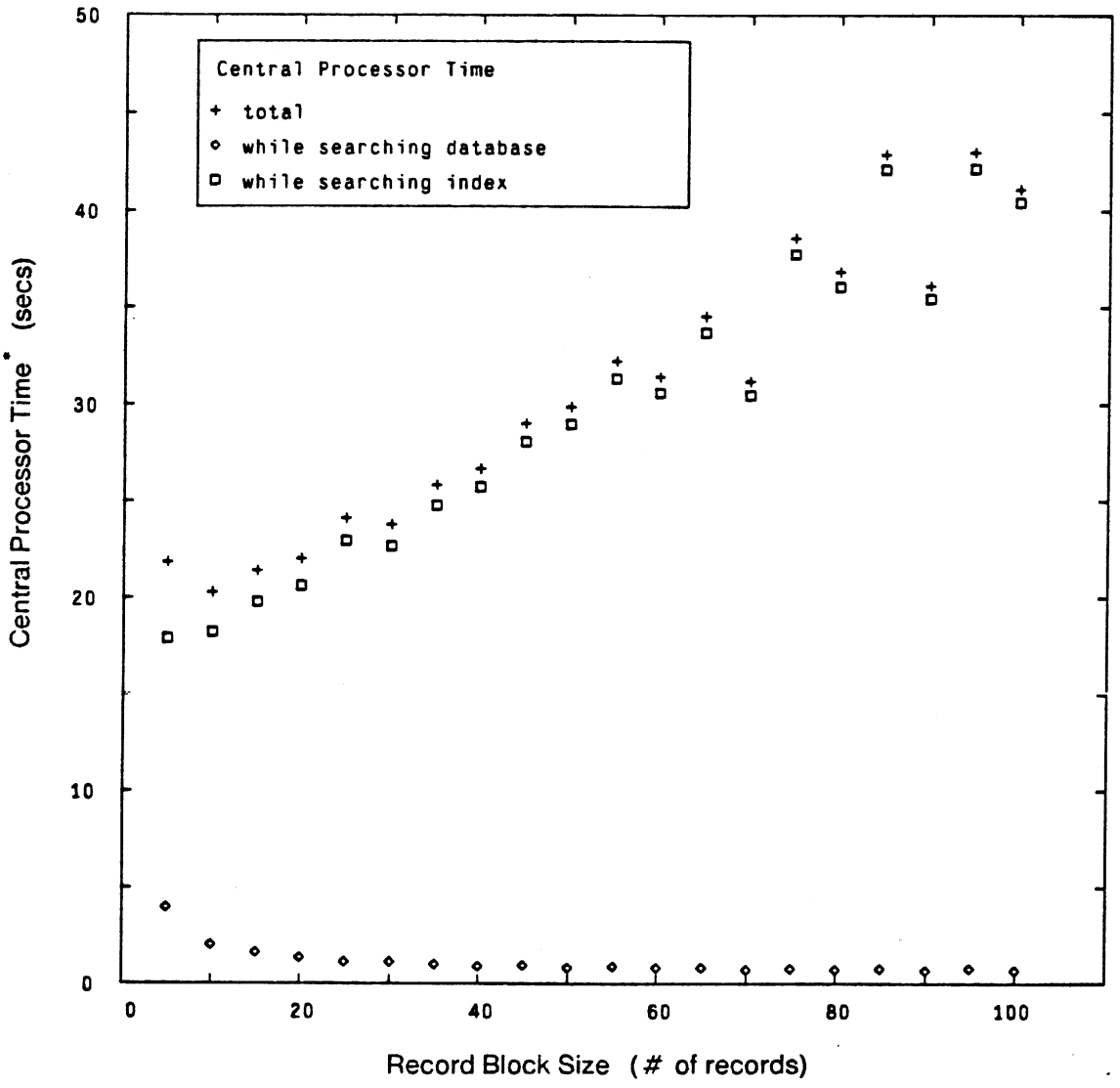
* for 200 names

Figure B-5: Disk Accesses vs. Record Block Size, Buffer Size 2048



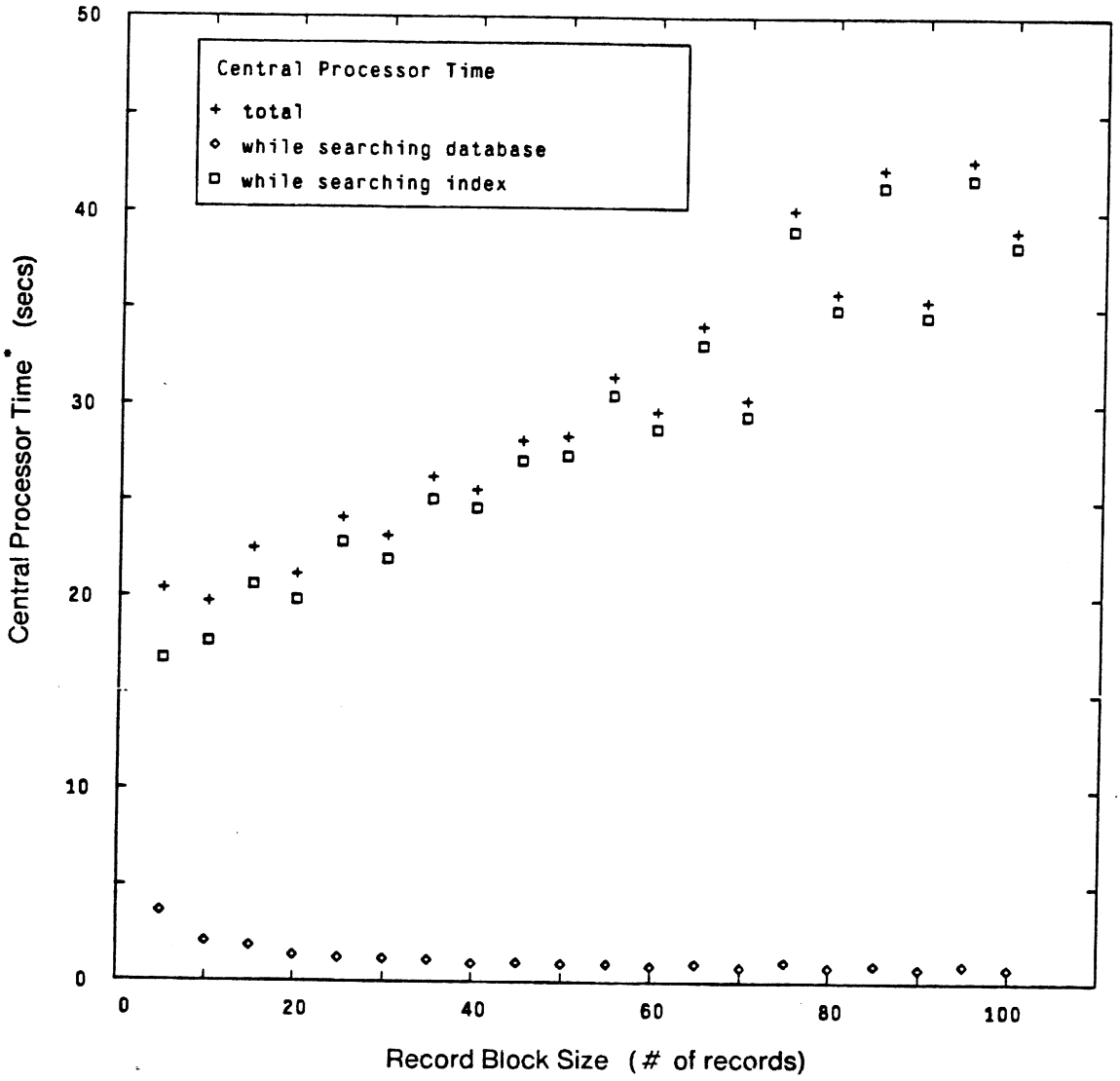
* for 200 names

Figure B-6: Central Processor Time vs. Record Block Size, Buffer Size 128



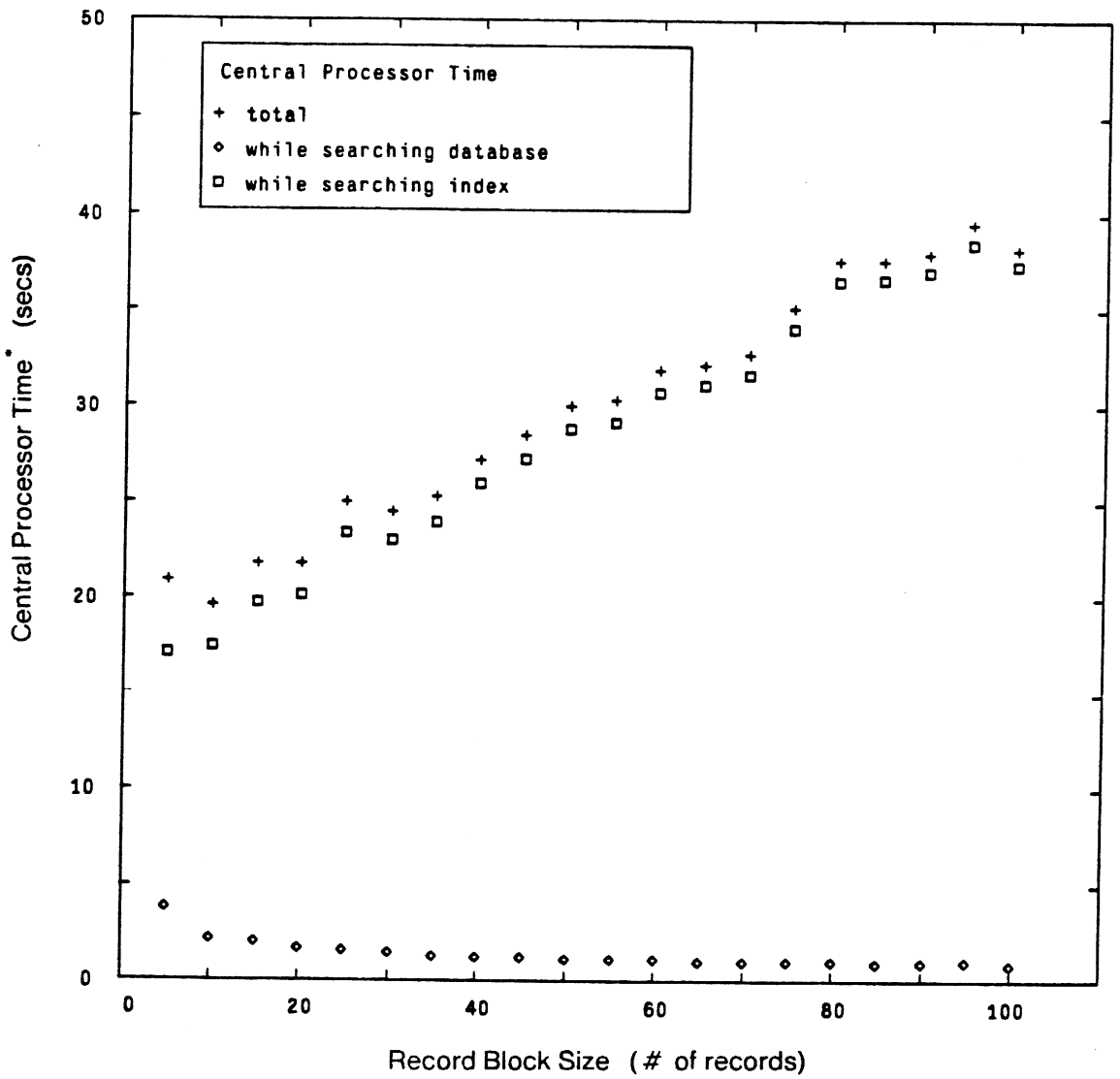
* for 200 names

Figure B-7: Central Processor Time vs. Record Block Size, Buffer Size 256



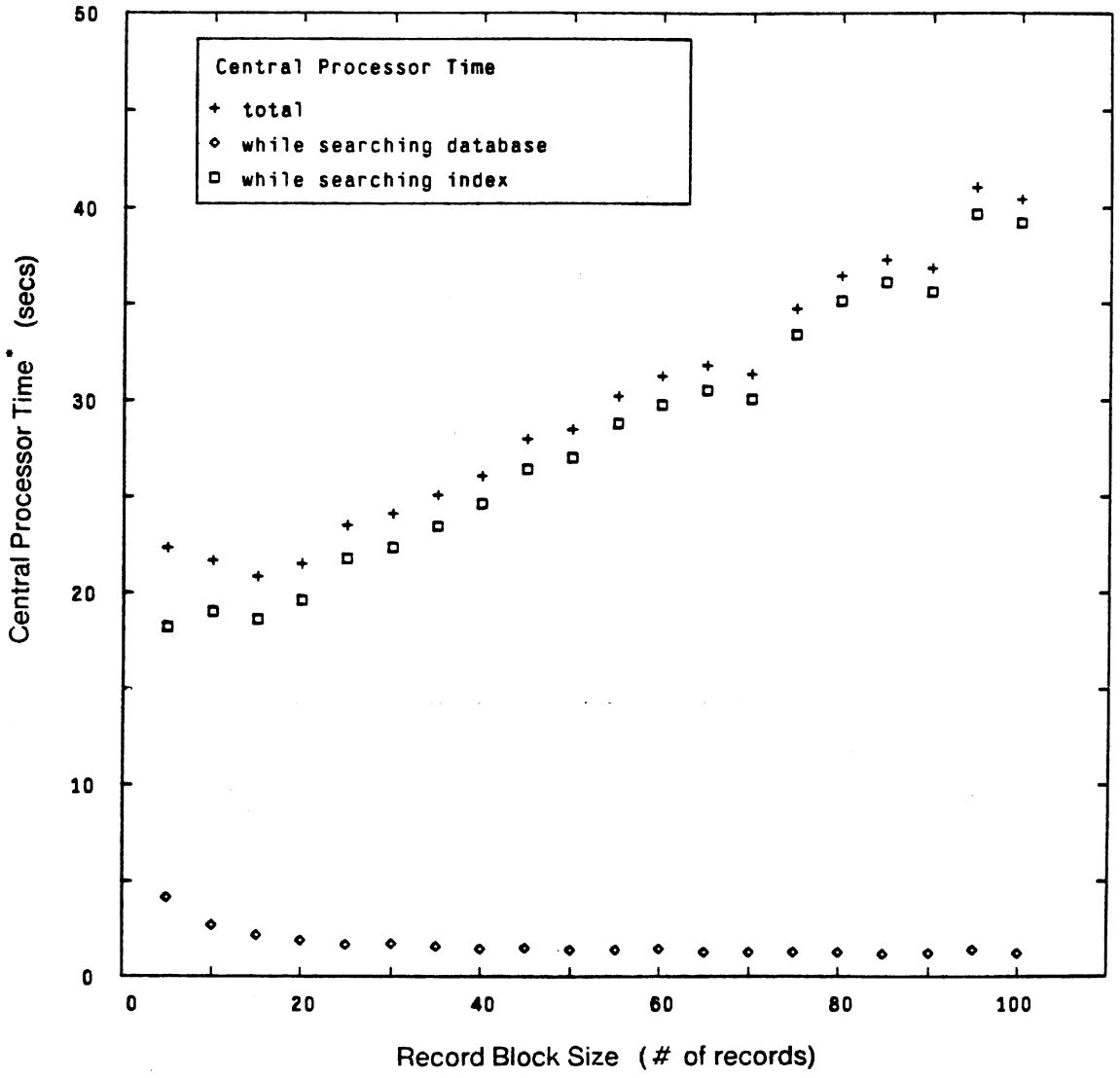
* for 200 names

Figure B-8: Central Processor Time vs. Record Block Size, Buffer Size 512



* for 200 names

Figure B-9: Central Processor Time vs. Record Block Size, Buffer Size 1024



* for 200 names

Figure B-10: Central Processor Time vs. Record Block Size, Buffer Size 2048

Bibliography

[1] Al-Awar, J., Chapanis, A., and Ford, W. R.
Tutorials for the First-Time Computer User.
IEEE Transactions on Professional Communication PC-24:30-37, March, 1981.

[2] Baker, J.D. and Goldstein, I.
Batch vs. Sequential Displays: Effects on Human Problem Solving.
Human Factors 8:225-235, 1966.

[3] Benbasat, I., Dexter, A. S., and Masulis, P. S.
An Experimental Study of the Human/Computer Interface.
Communications of the ACM 24(11):752-762, November, 1981.

[4] Bennett, John L.
The User Interface in Interactive Systems.
In Cuadra, C. A., editor, *Annual Review of Information Science and Technology*, pages 159-196. ASIS, Washington, D.C., 1972.

[5] Birrell, A. D., Levin R., Needham, R. M., and Schroeder, M. D.
Grapevine: An Exercise in Distributed Computing.
Communications of the ACM 25(4):260-274, April, 1982.

[6] Black, John B., and Moran, Thomas P.
Learning and Remembering Command Names.
In *Proceedings of the Human Factors in Computer Systems Conference*, pages 8-11. ACM, March, 1982.

[7] Black, John B. and Sebrechts, Marc M.
Facilitating Human-Computer Communication.
Applied Psycholinguistics 2(2):149-177, 1981.

[8] Borman, Lorraine and Karr, Rosemary.
Evaluating the Friendliness of a Timesharing System.
SIGSOC Bulletin :8-11, July, 1980.

[9] Butler, T. W.
Computer Response Time and User Performance.
In *Proceedings of the Human Factors in Computing Systems Conference*, pages 58-62. ACM, December, 1983.

[10] Davies, Donald W. and Yates, David M.
Human Factors in Display Terminal Procedures.
In *Proceedings of the Fourth International Conference on Computer Communication*, pages 777-783. International Council for Computer Communication, September, 1978.

[11] Edwards, Allen L.
Techniques of Attitude Scale Construction.
Appleton-Century-Crofts, New York, 1957.

[12] Edwards, Allen L.
A Technique for the Construction of Attitude Scales.
In Summers, Gene F., editor, *Attitude Measurement*, pages 214-221. Rand McNally, Chicago, Ill., 1970.

[13] Edwards, Ward.
The Theory of Decision Making.
Psychological Bulletin 51(4):380-417, 1954.

[14] Engel, Stephen E. and Granda, Richard E.
Guidelines for Man/Display Interfaces.
Technical Report 00.2720, IBM, April, 1975.

[15] Frierson, Elanor and Atherton, Pauline.
Survey of Attitudes Towards SUPARS.
In *Proceedings of the American Society for Information Science*, pages 65-69. ASIS, Greenwood, Westport, Conn., 1971.

[16] Gaines, Brian R.
The Technology of Interaction -- Dialogue Programming Rules.
International Journal of Man-Machine Studies (14):133-150, 1981.

[17] Gebhardt, Freidrich and Stellmacher, Imant.
Design Criteria for Documentation Retrieval Languages.
Journal of the American Society for Information Science 29:191-199, 1978.

[18] Good, Michael.
An Ease of Use Evaluation of an Integrated Editor and Formatter.
Technical Report TR-266, M.I.T. Laboratory for Computer Science, November, 1981.
Revised version of M.I.T. Master's Thesis

[19] Granda, Richard E.
Man/Machine Design Guidelines for the Use of Screen Display Terminals.
In *Proceedings of the Human Factors Society's 24th Annual Meeting*, pages 90-92. Human Factors Society, October, 1980.

[20] Harivel, J.
Use of a Dedicated Machine Within the Electronic Directory Project.
In Parslow, R.D., editor, *Information Technology for the Eighties*, pages 529-545. British Computer Society, Heyden, London, UK, July, 1981.

[21] Harrenstien, Ken and White, Vic.
NICNAME/WHOIS.
RFC 812, Network Information Center, SRI International, March, 1982.

[22] Heise, David R.
The Semantic Differential and Attitude Research.
In Summers, Gene F., editor, *Attitude Measurement*, pages 235-253. Rand McNally, Chicago, Ill., 1970.

[23] Hodge, M. H. and Pennington, F. M.
Some Studies of Word Abbreviation Behavior.
Journal of Experimental Psychology 98(2):350-361, 1973.

[24] Hsu, F. S.
Design of a Human Interface for an Online Directory Assistance System.
Bachelor's thesis, M.I.T. Dept. of Elec. Eng. and Comp. Sci., May, 1983.

[25] James, E.B.
The User Interface: How May We Compute.
In Coombs, M. J. and Alty, J. L., editors, *Computing Skills and the User Interface*, pages 337-371. Academic Press, New York, 1981.

[26] Knuth, Donald E.
The Art of Computer Programming. Volume 3: *Sorting and Searching.*
Addison-Wesley, Reading, Mass., 1973.

[27] Landweber, L., Litzkow, L., Neuhengen, D., and Solomon, M.
Architecture of the CSNET Name Server.
In *Proceedings of the Symposium on Data Communications*, pages 146-153. ACM SIGCOMM, March, 1983.

[28] Ledgard, Henry, Singer, Andrew, and Whiteside, John.
Directions in Human Factors for Interactive Systems.
Springer-Verlag, New York, 1981.

[29] Likert, Rensis.
A Technique ffor the Measurement of Attitudes.
Archives of Psychology 22:1-55, 1932.

[30] Liskov, Barbara, et al.
CLU Reference Manual.
Technical Report TR-225, M.I.T. Laboratory for Computer Science, October, 1979.

[31] Lucas, R. W.
A Study of Patients' Attitudes To Computer Interrogation.
International Journal of Man-Machine Studies 9:69-86, 1977.

[32] Maguire, Martin.
An Evaluation of Published Recommendations on the Design of Man-Computer Dialogues.
International Journal of Man-Machine Studies 16:237-261, 1982.

[33] Mann, J.
Decision Making.
The Free Press, New York, 1977.

[34] Miller, L. A. and Thomas, J. C.
Behavioral Issues in the Use of Interactive Systems.
International Journal of Man-Machine Studies 9(5):509-536, 1977.

[35] Miller, Robert B.
Response Time in Man-Computer Conversational Transactions.
In *Proceedings of the 1968 Joint Computer Conference*, pages 267-277. AFIPS, 1968.

[36] Miller, Robert B.
Human Ease of Use Criteria and Their Tradeoffs.
Technical Report 00.2185, IBM, April, 1971.

[37] Moran, Thomas P., editor.
"Special Issue: The Psychology of Human-Computer Interaction.
ACM Computing Surveys 13(1), March, 1981.

[38] Newman, William M. and Sproull, Robert F.
Principles of Interactive Computer Graphics.
McGraw-Hill, New York, 1979.

[39] Norman, Donald A.
Design Principles for Human-Computer Interfaces.
In *Proceedings of the Human Factors in Computing Systems Conference*, pages 1-10. ACM, December, 1983.

[40] Oppen, Derek C. and Dalal, Yogen K.
The Clearinghouse: A Decentralized Agent for Locating Named Objects in a Distributed Environment.
ACM Transactions on Office Information Systems 1(3):230-253, July, 1983.

[41] Osgood, Charles E., Suci, George J., and Tannenbaum, Percy H.
The Measurement of Meaning.
University of Illinois Press, Urbana, Ill., 1957.

[42] Osgood, Charles E., Suci, George J., and Tannenbaum, Percy H.
Attitude Measurement.
In Summers, Gene F., editor, *Attitude Measurement*, pages 227-234. Rand McNally, Chicago, Ill., 1970.

[43] Rayner, David.
Designing User Interfaces for Friendliness.
In Beech, David, editor, *Command Language Directions. Proceedings of the IFIP TC 2.7 Working Conference on Command Languages*, pages 233-242. IFIP, North-Holland, New York, September, 1980.

[44] Relles, Nathan and Price, Lynne A.
A User Interface for Online Assistance.
In *Proceedings of the Fifth International Conference on Software Engineering*, pages 400-408. IEEE, March, 1981.

[45] Roberts, Charles S.
Partial-Match Retrieval via the Method of Superimposed Codes.
Proceedings of the IEEE 67(12):1624-1642, December, 1979.

[46] Rouse, William B.
Systems Engineering Models of Human-Machine Interaction.
North Holland, New York, 1980.

[47] Shneiderman, Ben.
Software Psychology.
Winthrop, Cambridge, Mass., 1980.

[48] Solomon, M., Landweber, L., and Neuhengen, D.
The CSNET Name Server.
Computer Networks 6(3):161-172, July, 1982.

[49] Sondheimer, Norman K. and Relles, Nathan.
Human Factors and User Assistance in Interactive Computing.
IEEE Transactions on Systems, Man, and Cybernetics SMC-12(2):102-107, March/April, 1982.

[50] Spiliotopoulos, V. and Shackel, B.
Towards a Computer Interview Acceptable to the Naive User.
International Journal of Man-Machine Studies 14:77-90, January, 1981.

[51] Stallman, Richard M.
EMACS Manual for TWENEX Users.
AI Memo 555, M.I.T. Artificial Intelligence Laboratory, October, 1981.

[52] Teorey, Toby, J. and Fry, James P.
Design of Database Structures.
Prentice-Hall, Englewood Cliffs, NJ, 1982.

[53] Thurstone, L. L.
Attitudes Can Be Measured.
In Summers, Gene F., editor, *Attitude Measurement*, pages 127-141. Rand McNally, Chicago, 1970.

[54] Thurstone, L. L. and Chave, E. J.
The Measurement of Attitude.
University of Chicago Press, Chicago, Ill., 1929.

[55] Wasserman, Anthony I.
User Software Engineering and the Design of Interactive Systems.
In *Proceedings of the Fifth International Conference on Software Engineering*, pages 387-393. IEEE, March, 1981.

[56] Williams, C. W.
System Response Time: A Study of Users' Tolerance.
Technical Report 17-272, IBM, July, 1973.

CS-TR Scanning Project
Document Control Form

Date : 6/8/95

Report # LCS-TR-313

Each of the following should be identified by a checkmark:
Originating Department:

☐ Artificial Intellegence Laboratory (AI)
☒ Laboratory for Computer Science (LCS)

Document Type:

☒ Technical Report (TR) ☐ Technical Memo (TM)
☐ Other:_____

Document Information

Number of pages: 106 (113-IMAGES)
Not to include DOD forms, printer instructions, etc... original pages only.

Originals are:
☒ Single-sided or
☐ Double-sided

Intended to be printed as :
☐ Single-sided or
☒ Double-sided

Print type:
☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☐ Other:_____

Check each if included with document:

☒ DOD Form (2) ☐ Funding Agent Form ☐ Cover Page
☐ Spine ☒ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages (by page number): _____

Photographs/Tonal Material (by page number): _____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP (1-3) UN#'ED TITLE, ABST, ACK PAGES	
(4-106) PAGES #'ED 4-106	
(107-110) SCANCONTROL, PRINTER'S NOTES, DOD(2)	
(111-113) TRGT'S (3)	

Scanning Agent Signoff:
Date Received: 6/8/95 Date Scanned: 6/13/95 Date Returned: 6/22/95

Scanning Agent Signature: Michael W. Cook

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER MIT/LCS/TR-313	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* The Design and Implementation of an Online Directory Assistance System		5. TYPE OF REPORT & PERIOD COVERED M.S. Thesis January 1984 6. PERFORMING ORG. REPORT NUMBER MIT/LCS/TR-313
7. AUTHOR*(s)* Koile, Kimberle		8. CONTRACT OR GRANT NUMBER*(s)* DARPA/DOD N00014-75-C-0661
9. PERFORMING ORGANIZATION NAME AND ADDRESS MIT Laboratory for Computer Science 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS DARPA/DOD 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE March 1, 1984 13. NUMBER OF PAGES 106
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)* ONR/Department of the Navy Information Systems Program Arlington, VA 22217		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for Public Release, distribution is unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

Unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

directory assistance, name server, user interface

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This thesis describes the design and implementation of an on-line directory assistance system called DIRSYS that was modeled after the white pages of a paper telephone book and a full-screen display editor such as Emacs. As the user begins typing a name, the "pages" of this electronic telephone book appear on the screen, and the entry that most closely matches what the user has typed so far is highlighted. The system provides a tutorial for novice

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

20. continued.

users and an online help facility for novices as well as experienced users. The system also provides a facility for keeping the information in the database up-to-date. A preliminary evaluation of DIRSYS indicates that the system can be used easily by both inexperienced and experienced computer users and that, except for its slow performance, DIRSYS is usable and robust.

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 6/13/1995

M.I.T. Libraries
Document Services