1117

Bell Laboratories **Cover Sheet for Technical Memorandum**

The information contained herein is for the use of employees of Bell Laboratories and is not for publication (see GEI 13.9-3)

Title- System for Entering Data Through Computer Displayed Forms

Date- October 22, 1976

TM- 76-1352-6

Other Keywords- LSI-UNIX
CRT Terminal

Author(s)	Location and Room	Extension	Charging Case- 39394
E. W. Stark*			
H. Lycklama (Responsible Engineer)	MH 7C211	6170	Filing Case- 39394-11

ABSTRACT

This document describes two programs which make up a system for managing a database through the use of computer displayed forms. A form entry program displays a form on a CRT screen. The user types into the blanks on the screen and can either look up existing entries on the basis of the filled-in fields, or can store the displayed data as a new entry. In order that the form entry program be useful for a variety of displays, it reads in the display specifications from a file. The form entry program is thus completely independent of the data it is used with.

In order to facilitate the generation of new form displays and their specification files, a form editor program allows the user to manipulate the display directly on the CRT screen where it will be used.

One of the unique features of these programs is their use of program controlled buttons underneath the CRT screen. The buttons are used to perform cursor positioning and special program functions.

The form entry system runs on an LSI-11 computer under the LSX operating system.

*Johns Hopkins University

Pages Text 21 Other 3 Total 24

No. Figures 0 No. Tables 0 No. Refs. 7

Address Label

DISTRIBUTION
(REFER SEI 13.9-3)

VER SHEET ONLY TO	COVER SHEET ONLY TO	COVER SHEET ONLY TO	COVER SHEET ONLY TO
	CALLAHAN,R L	EPLEY,ROBERT V	HALPIN,T
	CANADAY,RUDD H	ERRICHIELLO,PHILIP M	HAMILTON,PATRICIA A
COVER SHEET ONLY TO	CANNON;T B	ESTES,VIRGINIA DANIELLE	HANSELL,P D
	CARDOZA,WAYNE M	ESTOCK,RICHARD G	HARRIS,CHARLES S
	CAREY,J E	EVENSON,E K	HART,R R
CORRESPONDENCE FILES	CARLSON,HELEN V	FARRELL,JAMES W	HAYDEN,DONALD F,JR
	CARPENTIER,A M	FARROW,LEONILDA A	HEDRICK,ELLEN L
4 COPIES PLUS ONE	CARRAN,JOHN H	FASCIANO,V	HELD,R W
COPY FOR EACH FILING	CARR,DAVID C	FEDER,J	HELLER,J ALLEN
CASE	CATO,H E	FEINBERG,HENRY R	HENDRICK,EDWARD J,JR
ES	CERRATO,R C	FERKER,NANCY L	HENIG,FRANCES H
ACKERMAN,A FRANK	CHAFFEE,N F	FIORE,RHODA J	HENRY,J B
ABO,ALFRED V	CHEE,T	<FLANAGAN,J L	HESS,B E
ALBERTS,BARBARA A	CHENOWETH,G R	FLANDRENA,R	HESS,MILTON S
ALLISON,C E,JR	CHERRY,LORINDA L	FLEISCHER,H I	HIRSCH,D
ALT,DOROTHY L	CHIANG,T C	FLEMING,J W	HOALST,B C
AMITAY,N	CHODROW,M M	FLYNN,MARY L	HODUM,HARRY C
AMOSS,JOHN J	CHRIST,C W,JR	FOLEY,G	HOOKER,J W
ANDERSON,C R	CLOUTIER,J	FONTENOT,SHARON M	HOPKIN,JAMES W,JR
ANDERSON,KATHRYN J	CLYMER,J C	FOO,YEOW PIN	HOUPT,WILLIAM R
RD ARNOLD,S L	COHEN,AARON S	FORTNEY,V J	HOYT,WILLIAM F
R ARTHURS,E	COHEN,HARVEY	FOUNTOUKIDIS,A	HO,DON T
C ARTIS,H P	COKE,ESTHER G	FOX,A	HO,TIEN-LIN
BERT M AVERY,J R	<COLE,LOUIS M	FRANK,H GREGORY	HUNT,C J
V AVIL,H J	<COLLIER,ROBERT J	FRANZ,ANN M	IERLEY,W H
CHAPIN BACCASH,JEANNE M	COLLINS,J P	FRASER,A G	IMAGNA,C P
IE T BAKER,BRENDA S	COOK,T J	FREEDMAN,MARVIN I	IPPOLITI,O D
W C <BALDWIN,GARY L	COPP,DAVID H	FREEMAN,R DON	IPPOLITO,N P
STAN L BALDWIN,GEORGE L	COREY,D A	FRENCH,WILLIAM G	IVIE,EVAN L
E,DEAN BATTAGLIA,FRANCES	COSTELLO,PETER E	FULCOMER,E J,JR	JACKOWSKI,D J
NO,PHILIP P <BAUGH,C R	COSTON,W P	<GATES,G W	JAKUBEK,R J
RY,S B BECK,R P	<COURTNEY-PRATT,J S	GELBER,CHERON L	JARVIS,JOHN F
LBARGER,DAVID BEL BRUNO,KATHLEEN A	CRAGUN,DONALD W	GERARD,A	JENSEN,PAUL D
AY,N B BENTON,JANET L	CRANE,R P	GERE,E A	JOHNSON,KEITH W
SE,A DICKSON BERGERON,DAVID N	CSASZAR,MARYANN	GERGOWITZ,E B	JOHNSON,STEPHEN C
IBER,J F BERNSTEIN,DANIELLE R	CUNNINGHAM,S J,JR	GEYLING,F T	JOHNSON,T W
EFFAUVER,W L BERTH,R P	DAVIDSON,CHARLES LEWIS	GIBSON,H T,JR	<JOHNSTON,W D,JR
EESE,W M <BEYER,JEAN-DAVID	DAVIS,D R	GIBSON,T A	JOHNSTON,W E,JR
KOGELNIK,H W BIANCHI,M H	DAVIS,R DREW	GILLON,ALEX C	JONES,B R
KUBIK,P S BICKFORD,NEIL B	DE GRAAF,D A	GILMER,G H	JONES,DAVID F
LAX,M BILLINGTON,MARJORIE J	DE JAGER,D S	GLASER,W A	JONES,M B
LIMB,JOHN O BIRCHALL,R H	DENNIS,T M	GLASSER,ALAN L	JONES,S E
LOZIER,JOHN C BIREN,IRMA B	DERMOND,FAITH L	GOETZ,FRANK M	JONES,VIRGINIA L
LUCKY,R W BLINN,J C	DEVLIN,N V	GOGUEN,N H	JONES,W MARTIN
<LYCKLAMA A NYEHOLT,H BONANNI,L E	DICKSON,MARY ANN T	GOLDENBERG,H R	<JUDICE,C N
MC DONALD,H S BOURNE,STEPHEN R	DOLOTTA,T A	GOLDSTEIN,A JAY	KAMINSKY,M L
MEIKI,A ZVI BOWRA,JAMES W	DONKIN,A E	<GORMAN,J E	KAPLAN,ALAN EDWARD
MILLER,STEWART E BOYLE,GERALD C	DOWDEN,IRIS S	GORTON,D R	KAPLAN,J J
NINKE,WILLIAM H BOYLE,J T,JR	DOWD,P G	<GRAHAM,R L	KAPLAN,MICHAEL M
PENZIAS,A A BRADLEY,M HELEN	DRENNING,C R	GRANDNER,W J	KARL,J W,3RD
PRIM,R C BRANDT,RICHARD B	DRUMMOND,R E	GRANGER,ROBERT H	KAUFELD,J C,JR
ROBERTS,CHARLES S BRANTLEY,JOAN T	DRYBURGH,PAULINE F	GRAVEMAN,R F	KAUFMAN,LARRY S
SLICHTER,W P BREEN,ROBERT N	DSTEFAN,D J	GREENBAUM,HOWARD J	KAYEL,R G
SWARTZWELDER,J C BROWNLOW,D L	DUDICK,ANTHONY L	GREENWALSH,H WAIN	KEATING,NANCY A
<TEWKSBURY,S K BROWN,C W	DUDLEY,ELIZABETH H	GRESHAM,J P	KENNEDY,ROBERT A
<THOMPSON,J S BROWN,JAMES W	DUNN,J C	GROSS,ARTHUR G	KERNIGHAN,BRIAN W
TUKEY,JOHN W BRUNO,BEVERLY A	DWYER,T J	GRUCHOWSKI,M P	KESSELL,D L
USAS,ALAN M BUCHANAN,D N E	DZIO,W H	GUIDI,PIER V	KETTERER,BARBARA A
WELLER,D R BURGESS,JOHN T,JR	EDELSON,DAVID	GUTT,DONALD J	KEVORKIAN,D E
WELLIVER,B S BURNETTE,W A	EDMUNDS,T W	HABNER,MRS IRENE A	KIKOLSKI,H J,JR
WOLONTIS,V MICHAEL BURNS,HARRY S	EITELBACH,DAVID L	HAIGHT,R C	KLAPMAN,RICHARD N
<YAMIN,ELAINE E BURROWS,THOMAS A	ELLIOTT,RUBY J	HALE,A L	KLEIN,RUTH L
50 NAMES	<BUTZIEN,PAUL E	ELY,T C	HALL,J T	KNAKKERGAARD,S F.
			...
			535 TOTAL

• NAMED BY AUTHOR > CITED AS REFERENCE < REQUESTED BY READER (NAMES WITHOUT PREFIX WERE SELECTED USING THE AUTHOR'S SUBJECT OR ORGANIZATIONAL SPECIFICATION AS GIVEN BELOW)

MERCURY SPECIFICATION.....

COMPLETE MEMO TO:
135-DPH 13-DIR 11-EXD 15-EXD 16-EXD 10-EXD 1352 1359 1356

COVER SHEET TO:
1271 1273 3114 8234 4145

UNED# • UNIX/EDITING AND DOCUMENT PREPARATION

LYCKLAMA,H
MH 7C211

TM-76-1352-6
TOTAL PAGES 26

TO GET A COMPLETE COPY:

1. BE SURE YOUR CORRECT ADDRESS IS GIVEN ON THE OTHER SIDE.
2. FOLD THIS SHEET IN HALF WITH THIS SIDE OUT AND STAPLE.
3. CIRCLE THE ADDRESS AT RIGHT. USE NO ENVELOPE.

PLEASE SEND A COMPLETE COPY TO THE ADDRESS SHOWN ON THE OTHER SIDE
NO ENVELOPE WILL BE NEEDED IF YOU SIMPLY STAPLE THIS COVER SHEET TO THE COMPLETE COPY.
IF COPIES ARE NO LONGER AVAILABLE PLEASE FORWARD THIS REQUEST TO THE CORRESPONDENCE FILES.

Bell Laboratories

Subject: System for Entering Data Through Computer Displayed Forms - Case 39394-11

date: October 22, 1976

from: E. W. Stark*

H. Lycklama
(Responsible Engineer

TM: 76-1352-6

MEMORANDUM FOR FILE

1. Introduction

In some applications it is necessary to manipulate a database which contains data that is structured such that each record is made up of a collection of parts, different records having similar data in corresponding parts. In this case, the records can be broken up into fields, each field labeled with an indication of its contents, and laid out in a two-dimensional arrangement. The result is a common data structure known as a form.

Computers are often used to manage such databases, however the data to be entered usually is written or typed on a paper form, and then transferred to the computer. The extra step can be eliminated, and the process of entering the data into the database eased by using a computer displayed form to replace the paper one. The programs described here form a system for managing a database through the use of forms

*Johns Hopkins University

displayed on a CRT screen. A data entry program allows the user to enter his data into the database, change or delete existing entries, or look up entries according to specifications typed in from his CRT display. A form editor program provides a means for manipulating the appearance of the form that the user sees as he stores his data into the database.

One of the unique features of the programs described here (FENTRY and MKFORM) is the use of buttons below the screen which are program controlled. These buttons perform commonly executed functions such as cursor positioning and clearing the screen, or special purpose functions unique to the program running. The buttons make it easy to place the cursor anywhere on the screen where data can be entered, and allow the user to correct errors.

Lookup and storage of records in the database is based on a key, which is a number of adjacent characters in the record. A hashing system is maintained in order to reduce the number of file accesses needed to retrieve a record. To lookup a record, the user must supply the key field; the program will then find the record and fill in the rest of the fields. Other options are provided to enable the user to lookup a record which matches certain characteristics, without having to type the key field.

Some CRT computer terminals, for example the Teletype Model 40 (1), contain hardware controls for creating 'protected' fields on the screen, that is, fields where the user

is prohibited from entering characters to the screen. The FENTRY data entry program described here also provides such 'protected' and 'unprotected' fields, however they are implemented in the form of software, rather than in the CRT terminal itself. A user may only position the cursor in one of the unprotected blank fields on the screen. These are the areas where data is entered. As the user types on the keyboard, the blank fields disappear and are replaced by the typed characters. Backspacing the cursor allows the user to correct typing errors. Typing over the end of a field causes a skip to the next field, letting the user type consecutive entries without worrying about the cursor position.

It is desirable to have a single, general-purpose program to manage databases and accept user input for a variety of form displays. Instead of making the data entry program dependent upon the form display, thus limiting the scope of application, the FENTRY program accepts descriptions of the database and form displays from disk files on the computer. Different form displays can be made up on the CRT screen through the use of the MKFORM program, then saved for later use as input to FENTRY.

Providing the underlying control for the form data entry system is the LSI-11 microcomputer. Although the LSI-11 could be used simply as an 'intelligent' terminal controller, in the present implementation, the LSI-11 not only provides the control for the CRT terminal, but the file

system support as well. The LSI-11 runs a stripped-down, single-user version of the UNIX operating system (2).

2. The LSI-11 UNIX System

The system used to run the form entry programs contains an LSI-11 microcomputer, 20K 16-bit words of dynamic MOS memory, a floppy diskette drive, and a CRT display. In addition to being able to run a complete single-user UNIX system, the system is connected to a PDP 11/45 computer running the UNIX-MERT (3,4) operating system. Large amounts of data can be loaded onto the floppy disk from the 11/45 fairly easily through this connection.

The UNIX system which runs on the LSI-11 is a single-user version of the larger time-sharing system (5). On the LSI-11, only 8K words of memory are required for the system routines, 12K words are left for a user program. The single-user system can support only three active processes at one time. Only the youngest process runs, and when it is complete, its parent runs, and so on. Another notable difference between the LSI-UNIX system and the standard system is the lack of in-core pipes. Even with these differences, however, the LSI-UNIX is still very useful, and is able to run most of the standard UNIX programs such as the C-compiler and the editor.

Since the floppy disk drives are rather slow, if a larger system with faster disks such as the 11/45 system is

available, it is advantageous to compile and edit programs on the 11/45 for later execution on the LSI-11. The execution of the programs on the LSI-11 is done with some routines on the 11/45 system known as the satellite processor system (6). With this system, programs actually run in the LSI-11 satellite processor, and system calls are sent back over the communications link to be executed on the 11/45. This is the way the form entry programs were debugged, although they can now run completely on the LSI-11 without support of the 11/45.

3. The CRT Terminal

The CRT terminal (7) used for the form programs will display 28 lines of 72 characters. Two memories are used, one to store the list of ASCII codes of characters to be displayed on the screen, and another to contain a set of character descriptions. The appearance of the characters can be changed by changing the prototypes stored in the second memory. Special scrolling modes allow the top line on the screen to be any line in the memory, thus eliminating the need for rewriting the whole memory when scrolling the display up one line. Cursor controls allow the program to specify the position and shape of the cursor.

Beneath the display screen there are sixteen user-programmable buttons which are used in the form entry system to implement cursor controls and other special functions. A

keyboard permits the user to type standard characters also.

4. Data Entry Program

The data entry program, designated FENTRY, contains code for accessing a database, displaying the form on the CRT, and receiving user input. The display is structured around blank fields, which appear on the screen as white rectangles. The cursor is prevented from moving anywhere except within one of the blank fields, and several special cursor control buttons are provided to facilitate movement of the cursor from one field to another. When a printing character is typed at the keyboard, it is placed in the blank field where the cursor was last positioned, and the cursor is moved to the next blank space to the right. If a field becomes filled, the cursor automatically skips to the next field to the right. In this manner, data is entered onto the screen until the blanks in the form have been filled.

In addition to the blank fields, the display also contains a number of labels which indicate what the contents of each of the fields should be when filled in. For example, a field which is intended to contain a person's name might have a label next to it which states 'NAME:'. The labels need not relate to the contents of the fields, however, nor even be in a one to one correspondence with them, as the labels have no significance to the data entry program except

as strings to be displayed on the screen.

When a user has filled in the fields on the screen he then selects one of the commands, depending on whether he is interested in looking up an entry, changing an entry, or storing new information in the database. The keyword for the command is typed into a special blank field on the screen which is reserved for this purpose, and the SEND button is depressed. Upon receiving the SEND signal, the program accepts the input from the screen, and checks to see if a legal command keyword has been typed. If so, the command is executed and if a new display is called for, new data will appear on the screen. Appropriate diagnostic messages appear on the screen above the special command field if anything goes wrong.

5. Form Composition/Editing Program

In order to make the data entry program really useful, it should not be limited to displaying one form, or to one database. The FENTRY program will accept input files which describe the database and the display to be generated. The user designs the form to be displayed with the MKFORM program.

The MKFORM program allows the user to use the CRT display and special buttons to move around characters and blank fields on the screen until a desired display is achieved. The program relieves the user of the task of

keeping track of row and column numbers on the screen, and lets him easily change the form display until it is both aesthetically pleasing and functional. The program also provides a means whereby an old form can be changed to accomodate new data without too much effort.

When running MKFORM, the user can either start a new form description from scratch, or he can modify an existing one. Characters and fields are moved around on the the screen through the use of the special buttons. Some buttons are programmed to move the cursor in four directions, others 'push' characters already on the screen around like a snowplow pushing snow. When the user is satisfied with the result, the END button causes the description to be encoded and written on a file which can be read by FENTRY.

6. Detailed Description - FENTRY

FENTRY stores and retrieves entries from the database through the use of a kind of hashing technique which should reduce the number of file accesses necessary to retrieve a certain entry. At any one time, there is a field which is designated the key field. The key field can be any contiguous sequence of blank spaces on the screen, and can even span boundaries between blank fields if the two fields immediately follow one another on the same line, or if the rightmost field on a line, and the leftmost field on the line immediately below are involved. This is due to the

fact that in the database a record is made up of all the blank spaces on the screen concatenated into one entry, without regard for the delimiting protected spaces which are displayed. As far as FENTRY is concerned, the key field consists of a starting offset into a record, and a size indicating how many characters after that offset make up the key.

As each entry is stored into the database, a number is computed from the characters in the key field by a fairly arbitrary algorithm. This number is used as an index into a table of pointers to information about entries whose hash matches the one for the entry to be stored. If there are no other entries at this point whose hash matches the current one, the entry is simply stored in the database, and a pointer to the information about the entry saved in the table. If there are already entries with the same hash number, the current entry is linked onto the end of a chain of entries with the same hash.

When an entry is looked up, the same procedure is followed, except for a slight difference in the case of entries with the same hash number. In addition to being identified by the hash number, each entry has a one-word identifier which consists of the first two bytes of the key field. If there are a number of entries with the same hash number, the one-word identifiers are compared until an entry is found that matches. Then the disk is accessed, and that entry is

read in. If the entire key field matches the one typed, the entry is displayed, otherwise, the program continues to try to match identifiers. If no matches are found, and the end of the linked chain is reached, a diagnostic message is displayed on the screen.

Two entries may exist with identical key fields, in which case when one of the entries is looked up, only the first one entered is displayed. The program remembers the entry displayed, and provides a function whereby the rest of the entries with the same key field can be displayed also. Since the one-word identifier consists of the first two bytes of the key field, with the first byte of the field in the most significant byte of the identifier, the identifier aids in alphabetization, which the program also does. As each entry is stored, a list is maintained which lists the pointers to the entries in alphabetical order. The 'alphabetical' ordering is done using the first characters of the key field as the most significant, each character being ordered in terms of its ASCII code. In order that a user could look up entries on the basis of a string which might be common to more than one entry (a last name, for instance) and alphabetize on a larger field, (the whole name) the alphabetization proceeds from the beginning of the key field until the number of characters compared is equal to another parameter, designated the alphabetical key size. Thus the user-settable key consists of an offset from the beginning

of each record, plus a size of the lookup key, plus an alphabetical key size.

6.1 Cursor Controls

In FENTRY, the cursor position is controlled through the use of ten buttons under the CRT screen. The last line on the screen is used to display labels for these buttons. In the present version, the labels appear as follows:

FST LST <-- <<- --> ->> UP DWN CLR SND

The FST button moves the cursor from wherever it is positioned to the beginning of the first blank field on the screen. The 'first' field in the following refers to a numbering of the fields starting from the upper left-hand corner of the screen, continuing across to the end of the line, over to the left-hand edge of the next line down, and so on. Similarly, the LST button moves the cursor to the last field on the screen. The last field is the command field, and is used solely for entering commands to FENTRY. Nothing typed in this field will appear as part of the record stored in the database.

The buttons labeled <-- and --> move the cursor one space to the left or one space to the right respectively. If the user attempts to move the cursor off of the edge of one of the blank fields, the cursor automatically skips to

the next field. This same feature is in effect if the user tries to type off the edge of one of the blank fields.

For movement of the cursor over larger distances on the screen, the following buttons are useful. The button labeled <<- jumps the cursor to the beginning of the previous field. If the cursor is presently in the interior of one of the fields, the cursor is positioned at the beginning of that field. Subsequent pushing of <<- result in jumps to the beginning of previous fields. Note that with any of the cursor control buttons, attempting to move the cursor off the screen has no effect. The ->> button is similar to the <<-, its function being to skip the cursor to the beginning of the immediately following field. If the cursor is positioned in the interior of a field, the ->> button will not position the cursor at the end of the current field, but at the beginning of the following field instead.

The UP and DWN buttons cause the cursor to move to the beginning of the first field on the first line above or below the current line which contains blank fields. The CLR button clears any diagnostic messages and all of the blank fields on the screen. The SND button is used when the user has filled in blank fields, entered a command into the special command field, and is ready to execute that command. The effect of SND is to cause the program to read in the contents of the screen, determine if the special command field contains a legal command keyword, and execute that

command if possible. Diagnostics appear if an error condition occurs.

6.2 Commands to FENTRY

There are seven commands to FENTRY: FIND, STORE, SEARCH, NEXT, DELETE, SETKEY, and EXIT. FENTRY reads in these commands from the command field on the screen when the SND button is depressed.

1. FIND

The FIND command causes FENTRY to read in the contents of the screen, hash on the key field, and look up an entry in the database whose key field matches the one typed. The record is displayed on the screen, and the command field is cleared, signalling the program's readiness to accept further commands or cursor control. If there is more than one entry in the database whose key field matches the one typed, only one of the entries is displayed. The other entries can be retrieved and displayed by the use of the NEXT command immediately after the FIND command.

2. STORE

Entering the STORE command causes the information currently displayed on the screen to be entered into the database. At present, no checks are made on the data to be entered, but a substantial improvement to

the program would involve a method of checking the data to be stored for consistency before allowing the information to be placed in the database. It would be useful to have these checks in a form similar to the form description files, so that the FENTRY program would not have to know anything about the data being entered.

3. SEARCH

The SEARCH command differs from the FIND command in that it lets the user type in arbitrary fields or parts of fields. The entire database is searched in alphabetical order for entries that match what was typed on the screen. In order for an entry to match the screen, the entry must have the same characters in spots where characters have been typed on the screen. A blank field matches anything, hence SEARCH'ing with a blank screen matches all of the entries in a database. After the first matching entry is retrieved, the command field is cleared. If the user wishes to continue to retrieve matching fields, he must enter the NEXT command into the command field and press SND. The NEXT command is never cleared from the command field and hence subsequent pressing of SND executes NEXT over and over again. A SEARCH or NEXT command when there are no matching entries left in the database has no apparent effect.

4. NEXT

The NEXT command is used in conjunction with the SEARCH or FIND command to retrieve all entries matching a typed in specification. The NEXT command has undefined effect when used without a preceding SEARCH, FIND or NEXT. When used after the FIND command, the NEXT command looks for another entry with a key field matching the one typed when the FIND command was sent. It is not necessary to retype the specification in order to execute repeated NEXT's, even though the specification may be cleared or replaced by other data on the screen.

When used after the SEARCH command, the NEXT command finds the next entry alphabetically in the database which matches the key field typed. When no more matching entries are left, the NEXT command does not produce a diagnostic, but simply has no effect. An execution of the NEXT command, unlike that of other commands, does not cause the command field on the screen to be cleared. This is so that repeated NEXT's can be sent without retyping the command field.

5. DELETE

The DELETE command causes an entry to be deleted from the database. In order to DELETE an entry, a FIND must be done to get the entry displayed on the screen.

This is so that entries with duplicate key fields do not cause conflicts. The DELETE command looks up the entry just as the FIND command does, but the entry must match completely, not just the key field as in the FIND command. If the entry is found, it is removed from the database, otherwise a diagnostic message is printed. If the FIND command is always used before the DELETE command, no diagnostics should ever be printed by DELETE.

6. SETKEY

The SETKEY command is used to set FENTRY's idea of the key field, and to change the hash tables accordingly. When SETKEY is typed in the command field and the SND button depressed, some instructions are printed on the line of the screen where diagnostic messages appear. The user should follow the directions, first placing the cursor at the beginning of the field to be used as the key, pressing SND, then at the end of the key field, pressing SND, and finally at the end of the field to be used as the alphabetization field, and pressing SND once more. The program then instructs the user to wait; if the database isn't very big the command won't take long. When the command field is cleared, FENTRY is ready to accept commands again.

SETKEY should be used when it is necessary to look

up a large number of entries by a certain field. If not too many entries need to be looked up, it is generally easier to use the SEARCH command than the SETKEY command. Note that after FENTRY creates a database, the key field is automatically set to the first field in the database, and the alphabetization field is the same as the key field. If the user wishes a different field for the key field, he should execute the SETKEY command.

7. EXIT

The EXIT command causes FENTRY to write out its internal tables into the database file, clear the CRT screen, and exit to the system level.

7. Detailed Description - MKFORM

The MKFORM program enables the user to design forms to be used with FENTRY. The forms are created by placing characters on the CRT and adjusting them with the control buttons until the desired form is achieved. The file produced by MKFORM is used as input to FENTRY and describes not only the screen display, but the size of each record in the database.

7.1 Cursor Control

When the MKFORM program is run, a button labeling scheme like that in FENTRY is used. The labeling display

appears as follows:

UP DWN <-- --> PUP PDN |-- --| CLR END

The UP, DWN, <--, and --> buttons move the cursor one space up, down, left and right, respectively. They can be used to position the cursor anywhere on the screen where characters can appear in the finished form. The cursor cannot be moved to the last four lines on the screen, since in the FENTRY program, they are used to hold diagnostics and the special command field.

The PUP, PDN, |--, and --| buttons are special buttons to ease movement of blocks of characters around the screen. In order for them to have any effect at all, the cursor must be positioned over a spot on the screen which contains either a character or a blank field. If this is done, when the buttons are depressed, the following actions will occur.

The PUP button causes MKFORM to determine the left and right-hand boundaries of characters contiguous to the one the cursor is positioned over. This set of contiguous characters on a single line is 'pushed' upward one line. In addition, if there are characters immediately above the contiguous set, they are pushed up one line also. If characters are pushed from the top line on the screen, they fall off the edge and are lost forever. The PDN button has essentially the same effect as the PUP button, except the

characters are pushed down one line, instead of up.

The --| button 'pushes' a set of contiguous characters which lie under and to the right of the current cursor position one space to the right. If characters go off the edge, they are lost. The |-- button works like --|, except the characters go left instead of right.

7.2 Creating a Form Description

To create a form description file with MKFORM, both labels and blank fields need to be placed on the screen. Labels are just ordinary characters, and are simply typed in from the keyboard onto the screen. To type a blank field character, type the DEL (rubout) key on the keyboard. This should cause a white rectangle to be displayed at the current cursor position.

After moving the labels and blank fields around on the screen and getting the screen the way you want it, press the END button. Incidentally, if you are editing a form description that you have no copies of, and you really mess it up, the ESC key (octal 033) can be typed and the program will exit immediately without rewriting your file. The CLR button clears all of the characters and blank fields off of the screen. It's only useful if you are really upset with the way the screen looks and want to start over.

Pressing the END button initiates three searches

through the screen. The first pass through finds all of the blank fields, and marks their positions. All searches occur from left to right, top to bottom of the CRT screen. Hence, the final structure of the record in the database will consist of all the blank spaces on the screen concatenated in this order. The second pass through the screen finds all of the labels and marks their positions. The third pass gathers up all of the strings and places them in the form description file.

8. Some Further Thoughts

The routine 'check' in FENTRY has been left blank. The original intention was to have this routine check an entry typed from the screen against the command to be run on it to inform the user if certain fields are not entered, or of nonsense fields, etc. In order to make the program apply generally to many different types of data, one routine could not be written to do specific checks on the data being entered with all forms. Possibly the user could create an object file with this routine specified according to his application, which could then be read in by FENTRY and executed just like part of the original code.

Other possibilities for investigation include the response time of FENTRY for large databases. The algorithm used to look up the entries was selected on the basis of how the author felt it would work, rather than on the basis of

any quantatative study. The time to look up an entry seems as though it would not depend upon the size of the entries, since pointers are kept in core so that the program can seek directly into the file at the proper point. It seems clear that response time will be very adequate for the size of the database which can be stored on a floppy diskette with approx 0.3M-bytes of storage. In any case, since the disk heads move rather slowly on the floppy drive, the database management system was designed for a minimum of head motion.

Hard copy of entries in the database would be useful. A program is needed to print out an entry or entries from the database in a useful format. Perhaps this feature could be made a part of the FENTRY program, implemented as a command e.g. PRINT.

MH-1352-EWS

E. W. Stark

Att.
References
Appendix A
Appendix B

References

1. Dept. 4145, Private communication about the use of the Teletype Model 40 in a Form Entry System.

2. H. Lycklama, "The LSI-UNIX System", TM-76-1352-4.

3. H. Lycklama and D. L. Bayer, "A Structured Operating System for the PDP-11/45", TM-75-1352-4.

4. D. L. Bayer and H. Lycklama, "MERT - A Multi-Environment Real-Time Operating System", TM-75-1352-7.

5. K. Thompson and D. M. Ritchie, "The UNIX Time-Sharing System", Comm. ACM 17, (July 1974), p365.

6. H. Lycklama and C. Christensen, "Emulation of UNIX on Peripheral Processors", TM-75-1352-2.

7. H. G. Alles, "A TV Terminal for the LSI-11 Micro-Computer", memo in preparation.

Appendix A
Structure of the Form Description File

The form description file is structured as follows:

First word: integer size (in bytes) of blank field information. No. of blank fields is this number divided by 6.

Second word: integer size (in bytes) of string position information. No. of strings is this divided by 6.

Third word: integer size (in bytes) of rest of file which contains strings.

Field info: Three words per entry, check 'fields' structure in fentry.c for meanings.

String info: Three words per entry, check 'strings' structure in fentry.c for meanings.

Strings: Ascii characters terminated by a null. Each null-terminated string corresponds in order to the string position information.

Appendix B
Compilation and Invocation of MKFORM and FENTRY

Bell Laboratories Murray Hill, New Jersey 07974 Compilation

Compilation of FENTRY and MKFORM depends upon whether they will be used with the LSI-UNIX stand-alone system or under 111 (satellite processor system). If they will be used with LSI-UNIX, just compile them on that system with

```
cc fentry.c

cc mkform.c
```

Or they may be compiled on a regular UNIX system, then relocated to begin at 40000(8).

```
cc -d -r fentry.c

reloc a.out 40000
```

If the programs will be used under 111, use the following as a prototype command sequence:

```
cc -c fentry.c

ldm fentry.o
```

Bell Laboratories Murray Hill, New Jersey 07974 Invocation and Options

There are no option flags at present with either of these programs. To run FENTRY, type:

fentry database descriptor

where 'database' is the name of the database file to use, and 'descriptor' is a file describing the form display (The output of MKFORM). If 'database' does not exist, it will be created. If 'descriptor' does not exist, a diagnostic message will be printed and the program will exit. Both arguments may be omitted, in which case the default names will be 'database' for the database, and 'formdesc' for the form descriptor.

The only way to invoke MKFORM is by the following:

mkform descriptor

If 'descriptor' does not exist, it will be created, otherwise it will be read in and displayed and editing will proceed from that point. If 'descriptor' does exist, it is not written until the END button is pushed. A temporary file is created instead of truncating the original.