Lawrence Berkeley National Laboratory
Recent Work

Title
USER'S GUIDE FOR THE TRAINING DATABASE SYSTEM

Permalink
https://escholarship.org/uc/item/6fm336jp

Author
Konrad, A.

Publication Date
1986-04-01

eScholarship.org Powered by the California Digital Library
University of California

PUB-3043 c.1

Lawrence Berkeley Laboratory

UNIVERSITY OF CALIFORNIA, BERKELEY

Information and Computing Sciences Division

User's Guide for the Training Database System

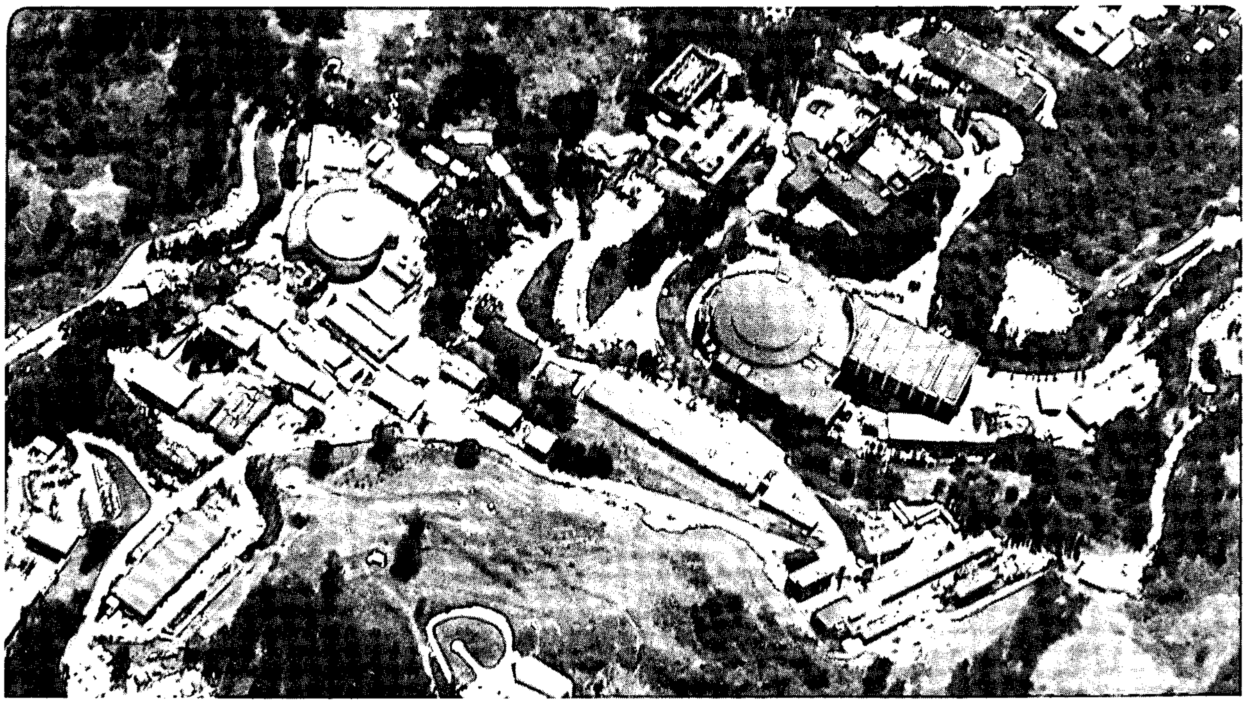
A. Konrad

April 1986

[illegible] 1988 [illegible] SECTION

For Reference

Not to be taken from this room



PUB-3043 c.1

Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.

DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

PUB-3043

User's Guide for the Training Database System

Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

Allan Konrad, X5458
Office of Computing Resources
Computing Division

Version 2.1
02 April 1986

Contents

I. Introduction:

I.1 Purpose

The LBL TRAINING database was implemented at the request of the Environmental Health and Safety Dept. for safety training courses. However, it is a generalized system that can serve the needs of general training administration as well.

The purpose of the TRAINING database system is to maintain current descriptive information about courses and students (whether LBL employees or not). It allows non-LBL employees to be included and provides data elements for off-site addresses.

Such data as student identification, course name, course date, and sponsoring agency is made available *without* requiring users of the TRAINING system also to maintain *employee information* such student name, mailstop, termination dates, payroll account number, etc. The intent is to alleviate the burden of having to maintain general employee information on users of specialized databases. Such unnecessary redundancy in employee-related databases at the Laboratory consumes an inordinate amount of maintenance effort. The TRAINING database system makes use of a *maintained* database of LBL employee information.

I.2 CMS, SPIRES, the CMS/SPIRES interface

The Stanford Public Information Retrieval System (SPIRES) is a product of Leland Stanford Junior University in Palo Alto, CA. The SPIRES database management system at LBL runs on the UC Berkeley Campus IBM 3081-D32 under the VM/CMS operating system. VM SPIRES consists of three components:

SPIRES itself (database management system)
CMS (the operating system that manages the computer)
SPIRES/CMS interface (maps SPIRES activity onto the CMS environment

Figure 1 indicates how these components relate to one another. Normally, SPIRES users are not and need not be concerned with the subsystems between themselves and SPIRES. The diagram is provided only to demonstrate context.

Most of the icons are self-explanatory. The purpose of the SERIES/1 is to make the user's ASCII terminal appear as an IBM 3270 terminal to the IBM 3081, and to make the IBM 3081 appear to communicate in ASCII to the user.

Section II will describe the commands to move along the path from terminal through the gateways into SPIRES. This generally requires less than 10 seconds and becomes routine.

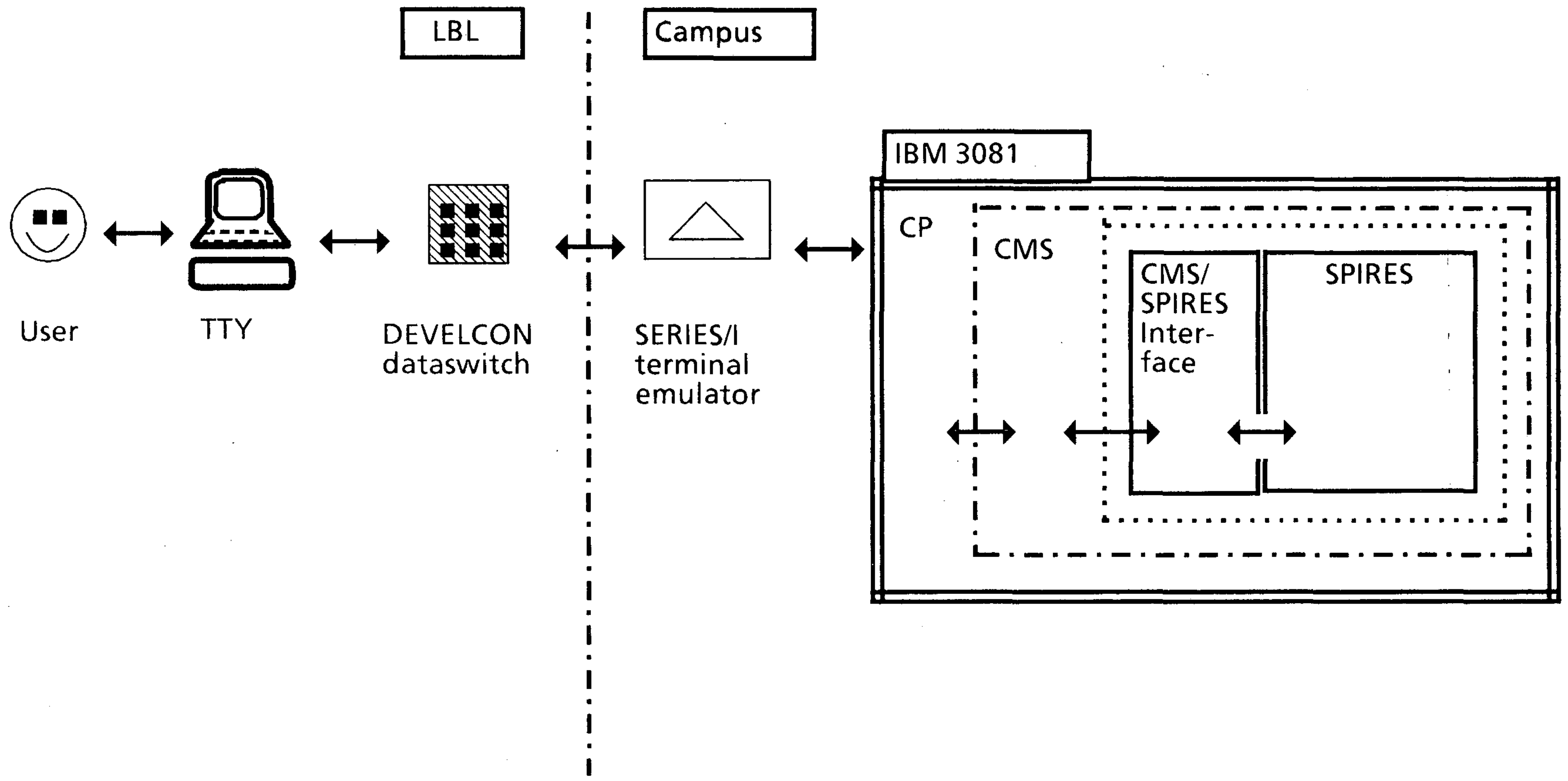


Figure 1. Pathway between user and SPIRES

I.3 General information about SPIRES.

Information for each course and student is stored directly into a SPIRES *record*. Each record in the database is comprised of a collection of elements as described below, e.g., course name, course date. The only student data required is the Employee ID (or similar code for non-LBL affiliated persons). Each record in a SPIRES database has a unique identifier often called *key*. In the COURSES database, the key is called **COURSE.CODE** (CC). In the STUDENTS database, the key of a record is called **ID**. Further explanation of keys used in the TRAINING system is found in Section IV.1 and V.1.

For each record, a particular element may be required or optional, singly or multiply occurring, have controlled allowable values, be limited to a particular type of value, and be indexed for ease in searching, etc.

If you are not in SPIRES, the CMS prompt is: **R;**

If you have EXITed SPIRES and you wish to re-enter, enter the command:

SPIRES

The normal SPIRES prompts are as follows:

-?	for UPPER case only
->	for upper and lower case
+?	UPPER case in Global For
+>	upper and lower case in Global For

All the modifications made to the database during the day (adds, updates, and removes) take effect immediately and are reflected the very next time the record is displayed. However, the indexes which are searched using the FINd command are not usually updated until early the next morning. Thus, FINd commands will not necessarily fetch records with newly added values for indexed elements until the next day (pending implementation of immediate indexing).

For most SPIRES commands, only the first three characters need be entered. For example, the **FIND** command requires only **FIN** <index> <value>. In of this document, commands will be fully spelled out, with the first three letters capitalized; e.g., FINd, SHOw ACTive, indicating that only the capitalized characters need be entered.

The term *file* as used in "TRAINING file" or "LBLSTAFF file", or "COURSES subfile" is distinct from *physical* CMS files and refers to SPIRES files, which are "logical" files that are physically stored in CMS files. The "active file" is also a SPIRES concept, and usually refers to the CMS file ACTIVE FILE A, Any CMS filename can be used as the SPIRES active file and is specifiable by the user with the SET ACtive <fn> <ft> <fm> command.

II. Getting Started.

II.1 Logging ON.

1. Turn terminal on and make sure the blue TSB box displays either a green or red light.

2. If red light is illuminated, press the blue button and wait for green light.

3. When green light is illuminated, enter carriage return [CR].

   The following dialogue should occur. The system reponse is in **bold**. The user response in modern font.

4. **Request:** ccdb [CR].

5. System will respond with a bell, and cursor and will jump to next line. Enter carriage return [CR].

6. **YALE ASCII TERMINAL COMMUNICATIONS SYSTEM V2.1**
   enter terminal type: adm3a [CR]. (or appropriate terminal type)

7. System will respond with a pseudo-three-dimensional display CFO over the letters VM. Enter another [CR].

8. The screen will clear. Enter:
   L [name of your virtual machine] [CR].

9. **ENTER PASSWORD:**
   enter your password. It is not a good idea to write your password in this set of instructions. If you write it down, do so elsewhere.

Note: If your previous session ended "abnormally", e.g., by simply pushing the blue button on the TSB box to obtain a red light, you will have to enter, at this point in the logon procedure, the command: **IPL CMS** and then a [CR]. This should always be done when a paragraph beginning with the word **"RECONNECTED . . ."** appears.

10. Enter yet another [CR]. This causes your PROFILE EXEC to execute. The system will then perform the following tasks automatically:

    call SPIRES
    SELect STUDENTS
    SET LENGTH 80
    SET UPLOW (for upper and lower case)

Note: Henceforth in this document, commands are assumed to be followed by a [CR], except for ESC-sequences and CNTL-sequences.

II.2 To LOGOFF

If you have one of the SPIRES prompts (-?, +?, ->, +>), enter: EXIT

The system will respond: **Leaving SPIRES.**

Enter: LOG

III. Organization of the database.

The LBL TRAINING database system is comprised of two *files*, each containing multiple *subfiles*, as shown in Figure 2.

The first file is SEALY82:TRAINING, which is itself made up of two component subfiles (databases): STUDENTS and COURSES. The TRAINING file may contain all of the data specific to training and courses at LBL as well as a number of indexes to make searching and retrieval easy.

The other file, TPHHH:LBLSTAFF contains all of the general employee information such as name, payroll account number, mailstop, building, room, extensions, and termination dates. This file serves as a centralized data source for several "satellite" applications similar to TRAINING. The SERVICE subfile is part of the LBLSTAFF subfile owned by the TPHHH virtual machine. The LBLSTAFF file is maintained by the Telephone Services Department in the Administration Division.

Each subfile is selectable as a database in its own right. The solid lines with two-way arrows in figure 2 indicate that data is shared between the two subfiles. The dotted line between TRAINING and SERVICES indicates that SERVICE is also used interactively for ID number verification. However, except for key elements, *data is never redundantly stored, but stored only in one database and then accessed by the others. This has the advantage that, when data is updated, it need only be updated in one place, yet this has the effect of updating all the user's data simultaneously.* Not only does this conserve staff effort, it assures consistency from database to database, from mailing list to mailing list.

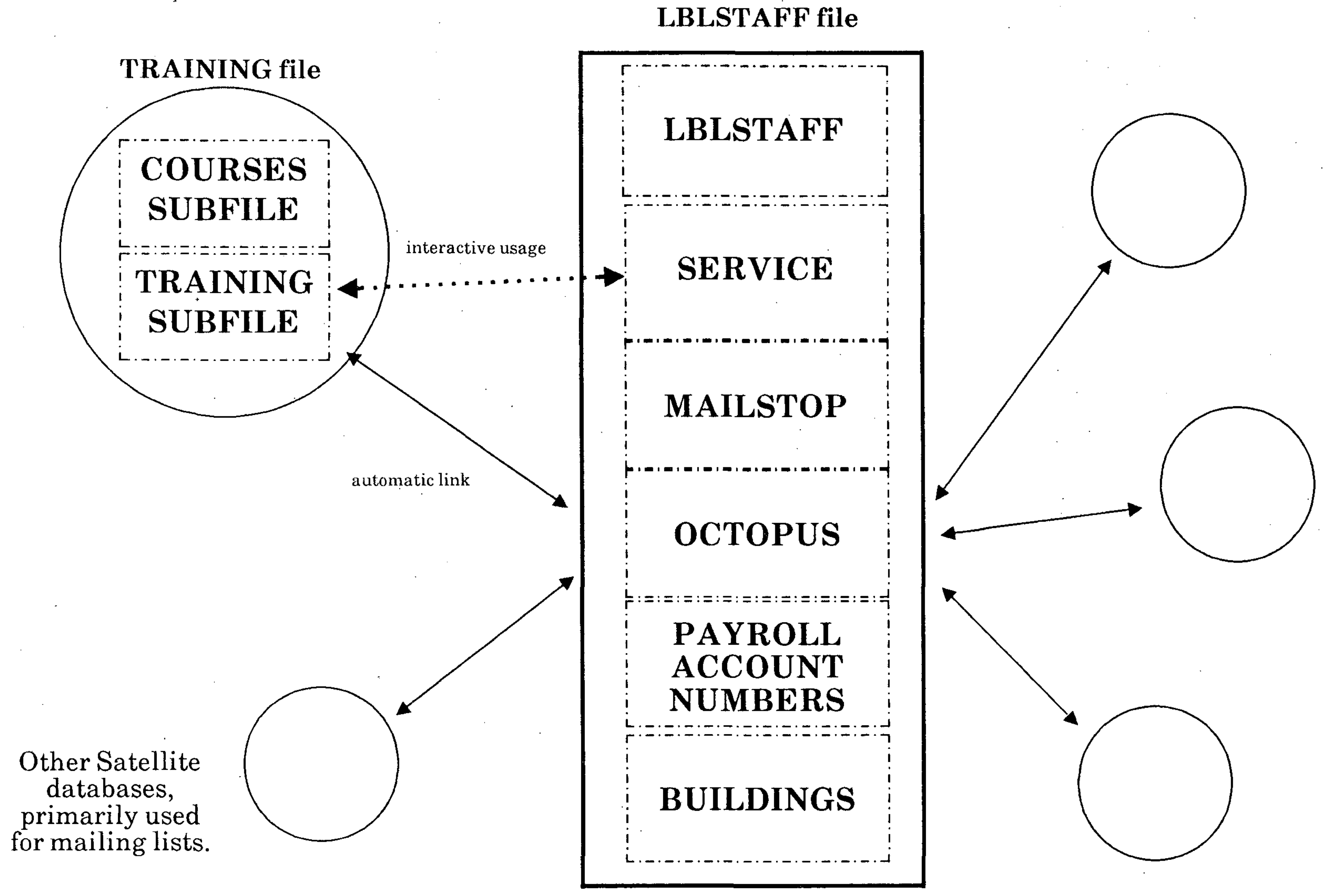

LBLSTAFF file
TRAINING file
COURSES SUBFILE
TRAINING SUBFILE
interactive usage
automatic link
LBLSTAFF
SERVICE
MAILSTOP
OCTOPUS
PAYROLL ACCOUNT NUMBERS
BUILDINGS
Other Satellite databases, primarily used for mailing lists.

Figure 2

This same "single source" concept is applied within the TRAINING database as well. To avoid redundant storage of data, separate subfiles contain information that may apply to more than one student. For example, since any particular course will have more than one student, the COURSES subfile contains most of the pertinent information about a course, rather than it being duplicated in full in the record for every student who has taken the course.

It is not mandatory that an indivdual be represented in the LBLSTAFF database to be included in the TRAINING database. The STUDENTS subfile provides for optional elements to store generalized information about off-site persons ("ALIENS") who may participate in LBL courses.

As mentioned, TRAINING file is composed of two subfiles (databases):
the STUDENTS subfile which contains data about students
the COURSES subfile which contains data about classes.

Each is selectable separately and interlinked with the other. The STUDENTS subfile can be selected for information about what courses a student has taken, the COURSES subfile can be used to find what students have participated in a course.

Each record in each subfile is comprised of *elements*. The elements and their attributes will be described below. Among these elements, each STUDENT record contains references to the courses with which that student is affiliated. Students are assigned to zero or more courses, i.e., a student record can exist without being affiliated with a course (since the database is used for other purposes). The COURSES subfile acts as a lookup table, as well as a database, to insure that the courses listed in student records are valid.

The TRAINING database also makes use of SPIRES privilege-tailoring. That is, only certain users may be allowed to see and/or update certain elements. SPIRES prohibits users from deleting a record if the record has elements on which the user does not have privileges to see or update. The database owner can see all the elements and will remove the record.

Certain data validation is provided. For example, an error is reported if an attempt is made to add a course to a student's record if that course has not yet been added to the COURSES subfile.

SPIRES provides for owner-defined charging to enable recovery and sharing of costs of storage and maintenance of the system. Charging can be implemented on an hourly basis, by the number of records displayed, or other criteria. Any such charges will be announced if and when they are implemented.

The TRAINING database system contains several virtual elements. These are indicated in the listings of elements for each subfile (Please see IV.1 and V.1)

Virtual elements are elements which do not exist but appear to exist. That is, they are not stored and they cannot be edited. They *can* be displayed, records *can be* sequenced according to their values, and indexes *can be* built based on their values. Generally, they are derived from elements in other subfiles or other databases, or system information.

By default, only the real elements are displayed in SPIRES. To be able to see the virtual elements, a SET ELEM command must be issued in SPIRES. However, for the TRAINING application, the SET ELEM command is automatically issued each time the STUDENTS Subfile is SELected. If you wish *not* to see them, you may issue the command CLR ELEM. The command SHOw ELEM will indicate which virtual elements are to be displayed by notating " - SET" after the element name. If you clear away the virtual elements with CLR ELEM, you can reset them easily by the command SETELEM. Notice that SETELEM has no space between the words, but CLR ELEM does.

The TRAINING system provides a variety of indexes to pertinent information. Some of the indexes index data that does not exist in the TRAINING system. For example, employee payroll account numbers do not exist as stored information, yet they can be searched in real indexes in the STUDENTS subfile. This is accomplished by use of SPIRES virtual elements which have length of zero and are generated from other elements rather than stored.

Occasionally you may wish to use some other format than the default SPIRES format. Customized formats have been provided, as described in section. The command SHOw FORmats will list these and also indicate if either one is currently in effect by notating " - SET" after the format name.

RULE: If any virtual element is **SET**, then the format will not take effect. That is, presence of a virtual element overrides a format. Thus, in order to use a format, you should enter the command **CLR ELEM** beforehand. The REPORT EXEC does this automatically for making reports.

The same restriction applies to generating tables with the **DEFine TABle** or the **SET FORmat $REPORT** formats. Virtual elements must be cleared by **CLR ELEM** before the table will take effect.

IV. Using the COURSES Subfile

The purpose of the COURSES subfile is to maintain current descriptive information about each course in a single place so that it does not have to be redundantly copied into each student's record who participated in that course. **A course record must be added to the COURSES subfile before a student's record can refer to that course.**

Each course is represented in the database by a collection of elements as described below, e.g., sponsor, course title, etc. Each record has a unique identifier often called *key* or in the COURSES database, **COURSE.CODE** (CC). This code is assigned by by whoever enters the new course record into the database. The the convention is to constuct the code as follows:

<dept> - <number>

For example,

EHS-0115

refers to a course in the Environmental Health & Safety Department.

For each record, a particular element may be required or optional, singly or multiply occurring, have controlled allowable values, be limited to a particular type of value, or be indexed for ease in searching, etc. The element listing below describes the characteristics of each element.

IV.1 Description of elements in the COURSES subfile

Element Name	Required/Opt	Length	Occurrences	Data Type	Indexed
COURSE.CODE (CC) (key of the record)	Required	Variable	Single	Character	Immed. Index
COURSE.TITLE (CT)	Required	Variable	Single	Character	Immed. Index
POINTER (noupdate/nosee)	Optional	Fixed	Multiple	Hex	
HOURS	Optional	Variable	Single	Character	
SPONSOR.NAME (SN)	Optional	Variable	Single	Character	Immed. Index
COURSE.DATE (CD)	Optional	Fixed	Single	Date	
INSTRUCTOR (I)	Optional	Variable	Multiple	Personal Name	Immed. Index
NOTE.STR	Optional	Variable	Multiple	Structure	
NOTE	Optional	Variable	Single	Character	
NOTE-DATE	Automatic	Fixed	Single	Date	
COURSE.CATEGORY (CCAT)	Optional	Variable	Single	Character	
DATE.UPD	Required (automatic)	Fixed	Single	Date	
COURSE.COST (COST)	Optional	4	Single	Dollar	
FUNDING.SOURCE (FS)	Optional	Variable	Multiple	Character	
Student data			Single	Phantom Structure	

Elements which are indexed are searchable using the FINd command, described below

Graphically, the hierarchical nature of a typical COURSES record appears:

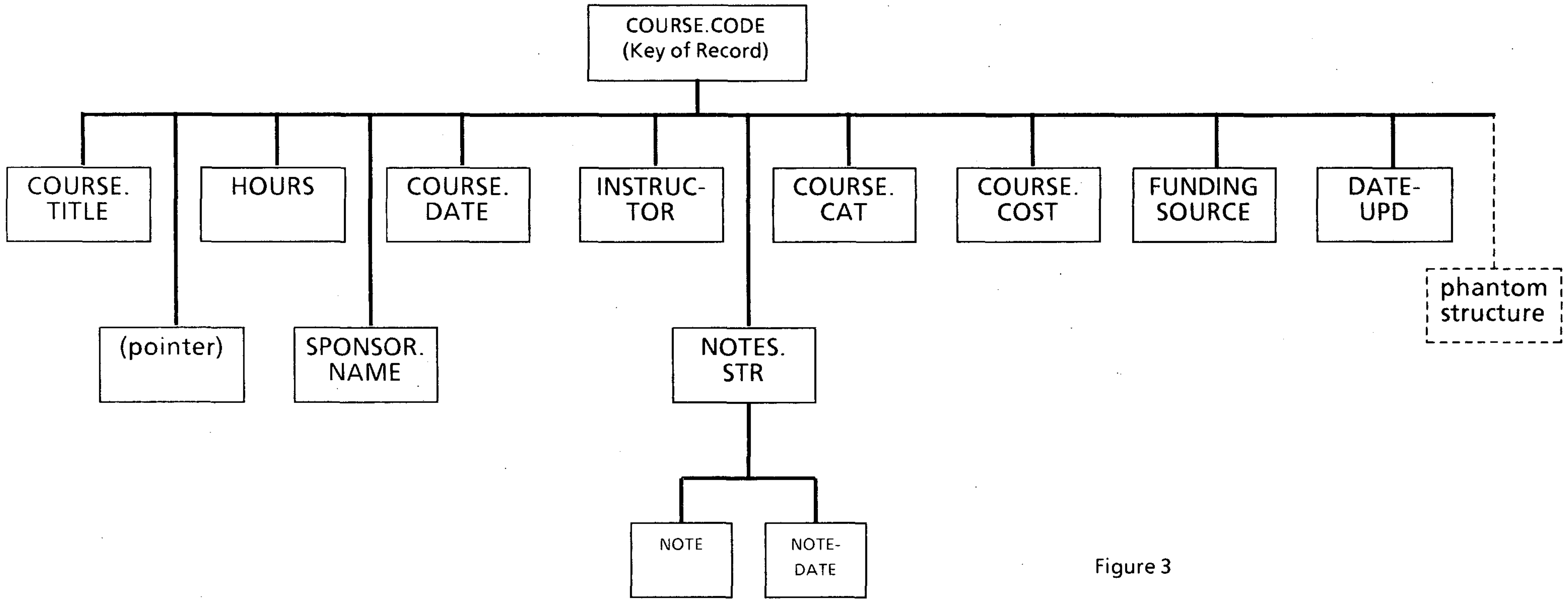

COURSE.CODE
(Key of Record)
COURSE. TITLE
HOURS
COURSE. DATE
INSTRUC- TOR
COURSE. CAT
COURSE. COST
FUNDING SOURCE
DATE- UPD
phantom structure
(pointer)
SPONSOR. NAME
NOTES. STR
NOTE
NOTE- DATE

Figure 3

IV.2 Displaying Records.

To use the COURSES subfile, you must SELect it with the command:

SELect COURSES

If you know the COURSE.CODE for a record which you wish to see, you may use the DISplay command to view it directly:

DISplay <COURSE.CODE>

or, by using the FINd command, and TYPe:

FINd <COURSE.CODE>
TYPe

If you do not know the COURSE.CODE for a record which you wish to see, then you must search for it based upon some criteria you do know. Use the FINd command to search for records in this way (Section IV.3) Then, to look at the records which are the result of a **FINd** command, enter the command **TYPe**. All of the records in the search result will then be displayed.

If you search on a non-indexed element (See Sect IV.3) i.e., using Global For, then use the **DISplay** <all/first/last/n/ext> command.

IV.3 Searching in SPIRES; Searching the COURSES Subfile

You may search for COURSES records based on any element or combination of elements. However, some elements are used as the basis of searching much more often than others. Those elements are *indexed* in the same way as selected keywords are indexed in the back of a book. Rather than searching sequentially through a book to find a particular topic, you find the topic in the index. Associated with its entry is an *address*, usually a page number. SPIRES indexes work in much the same way. Indexed elements are listed along with their "addresses". However, you never have to worry about the addresses. You simply enter a FINd command, and SPIRES fetches the addresses and then allows you to display, re-sequence, or update the records as desired.

To see a list of the elements in the subfile, enter the command **SHOw ELEMents.**

To see a list of indexes, enter the command **SHOw INDexes.**

As indicated in the element list above, those elements which are indexed are: COURSE.CODE (key of the record), COURSE.TITLE, INSTRUCTOR, and SPONSOR.NAME

The key of a record may also be searched as if it were an indexed element (with the FINd command), which, as implemented in SPIRES, it is, since goal records are stored in order by key.

To search for courses based on any of these elements, use the FIND command, as follows:

FIND <index name> <relational operator> <value>

For example, to find all the first aid courses, enter:

FIND COURSE = FIRST AID

Then use the TYPe command to see the result (Section IV.2).

If you do not include the relational operator in your search, SPIRES assumes an "equals" operator:

FIND COURSE FIRST AID

Less commonly-used elements are not indexed, for example, COURSE.DATE. To search for all the courses with a specific course date, enter

FOR TREe WHEre CD = 'mm/dd/yy'

For example,

FOR TRE WHE CD = '08/01/83'

Then use the DISplay command to see the result (Section IV.2).

It is permissible to use the FINd command to find the key of a COURSES record as described in Section IV.2.

Indexes are now updated immediately when a record containing indexed elements is updated. It is not necessary to wait for overnight processing for the indexes to reflect record updates. The FINd command will find a record immediately after an UPDate command is issued.

A complete description of all the searching capabilities in SPIRES is described in the document Searching and Updating listed in Appendix E.

IV.4 Updating Records.

A complete description of updating records in SPIRES is described in the document Searching and Updating listed in Appendix E. This brief summary provides an overview.

To update a COURSES record, enter the following commands:

1. Use the FINd and TYPe commands to determine the key of the record you wish to modify.
2. **TRAnsfer** <COURSE.CODE> **CLR**
3. **X ACTIVE FILE** (this enables you to use the editor to modify the file)
4. Edit the record. When all the changes are made, enter the command **FILE** on the command line. This will return you to SPIRES.
5. **UPDate**
6. **DISplay** <entry number> to verify that the record is correct.

When adding or modifying data elements, remember that the format is:

DATA ELEMENT = value ;

Don't forget the semicolon!

Further, if adding a note, insert the following lines:

NOTE.STR;
NOTE = <text of note, no limit to length> ;

Don't forget the semicolons!

SPIRES validates the data when you update the record. If there are any illegal values, you will receive an error message when you enter the **UPDate** command. If this occurs, return to step 3 and re-edit the record.

Finally, it is always a good idea to retain your source documents after you complete any updating. In four years of running SPIRES at LBL, no data has ever been lost, but users have forgotten why they changed some records.

Note that the current version of the record will always be displayed by SPIRES.

Indexes are now updated immediately when a record containing indexed elements is updated. It is not necessary to wait for overnight processing for the indexes to reflect record updates. The FINd command will find a record immediately after an UPDate command is issued.

NOTE: The key of a record (COURSE.CODE) **cannot** be modified by editing its value and the issuing an **UPDate** command. To change the key of a record, please see Section IV.7.

IV.5 Adding a new record.

A complete description of adding records in SPIRES is described in the document Searching and Updating listed in Appendix E. Below, a very brief summary and sample session provides an adequate overview.

SET FORMAT $PROMPT

ADD

You will be prompted for the value of each element. If an *optional* element should be left blank, enter a carriage return [CR]. Also note that you will be prompted twice for each multiply occuring element. Just enter a [CR] to proceed to the next element. Please see Appendix C for the subcommands used in the $PROMPT format.

To add several records, simply reissue the ADD command after each previous record is ADDed and DISplayed.

After ADDing a new record, always DISplay <key> to examine it for correctness. If a record was entered with an ID that does not match an ID in the LBLSTAFF subfile, then no employee information will appear when the record is displayed. Further, sometimes employee ID's change, especially when a person changes from guest to employee status. Thus, the ID used in a newly added STUDENTS record may not match that in LBLSTAFF. This can be resolved by searching the SERVICE subfile of the LBLSTAFF file on the employee's name (current or former) (Pending implementation of an exec to assist in entry and verification of data).

Here's a sample session showing how to add a record (system responses in **bold**):

```
sel courses (not necessary if COURSES is alreaded SELected)
set format $prompt
-?
ADD
:COURSE.CODE
EHS.0115  [CR]
:COURSE.TITLE
First Aid  [CR]
:HOURS
3   [CR]
:SPONSOR.NAME
:e&hs  [CR]
:COURSE.DATE
May 3, 1984.  [CR]
:INSTRUCTOR
Dr. Rocker   [CR]
   Struc:  NOTE.STR(1)
:  NOTE
This class required a $10 materials fee.  [CR]
:  NOTE-DATE
[CR]
   Struc:  NOTE.STR(2)
:  NOTE
[CR]
:DATE-UPD
[CR]
-?
dis EHS.0115   (or:   /dis $key)
```

```
COURSE.CODE = EHS.0115;
COURSE.TITLE = FIRST AID;
HOURS = 3;
SPONSOR.NAME = e%hs;
COURSE.DATE = Tuesday, MAY 3, 1984;
INSTRUCTOR = Dr. Rocker;
NOTES.STR;
  NOTE = This class required a $10 materials fee.;
  NOTE-DATE = 04/14/84;
DATE-UPD = 04/14/84;
```

IV.6 Removing Records.

A complete description of REMoving records in SPIRES is described in the document Searching and Updating listed in Appendix E. Below, a very brief summary and sample session provides an overview.

To REMove a record from the COURSES subfile, enter the command:

REM <COURSE.CODE>

For example, to REMove record EHS.0115, enter:

REM EHS.0115

Note: If you wish to restore a record that was erroneously removed, please call for assistance. The DEQueue command is disabled in this subfile.

NOTE: You will not be permitted by SPIRES to remove any record that contains a POINTER element. If you wish to remove such a record, you must first SELect STUDENTS, FIND the student records that have that COURSE.CODE, and modify them to exclude the course to be removed. The next day, you may then REMove the record in the COURSES subfile. If you attempt a REMove, and the system responds:

-PRIVILEGED COMMAND

then that course record has POINTERs. Follow the the procedure below:

Step 1. **SELect STUDENTS**
Step 2. **FINd CC** <COURSE.CODE of course to be removed>
Step 3. If the result is **not zero** then delete from each STUDENT record in the result the COURSE.CODE that is to be discontinued.

Step 4. The *following day* you may then **SELect COURSES**
Step 5. **REMove** <COURSE.CODE>

If you have verified that no records exist in the STUDENT subfile which refer to the course to be removed, and you still received **PRIVILEGED COMMAND**, then TRAnsfer and Xedit the record to add an occurrence of the NOTES element In the note, indicate that the record should be removed any why, along with your name and extension. The database owner will periodically scan these to accomplish the removal.

IV.7 Changing the Key of a Record.

The key of a record (COURSE.CODE) **cannot** be modified by editing its value and then issuing an **UPDate** command. To change the key of a record, enter the following commands:

1. create the new COURSES record (Sect IV.5)
2. SELect STUDENTS
3. FIND CC <old course.code> and update those records with the new COURSE.CODE (you may use FOR RESult, and successive TRA's and UPD's or a batch method)
4. The *following day* you may then SEL COURSES
5. **REMove** <key of the old COURSE record> (as described in Sect. IV.6 above, if permitted.)

Please ask for human help if you encounter any difficulties (Appendix F).

V. Using the STUDENTS subfile

This subfile is also selectable as TRAINING by the owner account.

Not all elements listed will necessarily be displayable or updatable by non-owner accounts.

The purpose of the STUDENTS subfile is to maintain current descriptive information about each student.

For each record, a particular element may be required or optional, singly or multiply occurring, have controlled allowable values, be limited to a particular type of value, or be indexed for ease in searching, **etc.** The element listing below describes the characteristics of each element.

The only element that is required to occur is the key, **ID**. If the individual does not have an LBL employee ID, then you may invent a 7-character ID. In the TRAINING and LBLSTAFF systems, LBL ID's must be 7 characters, the letter E followed by 6 natural numbers for employees, or the letters GU followed by 5 natural numbers for guests:

E999999 or **GU99999**

Usually a student record will have at least on occurrence of COURSE.STR (i.e., one course), though it is not mandatory. Some "students" only participate in Laboratory safety activities.

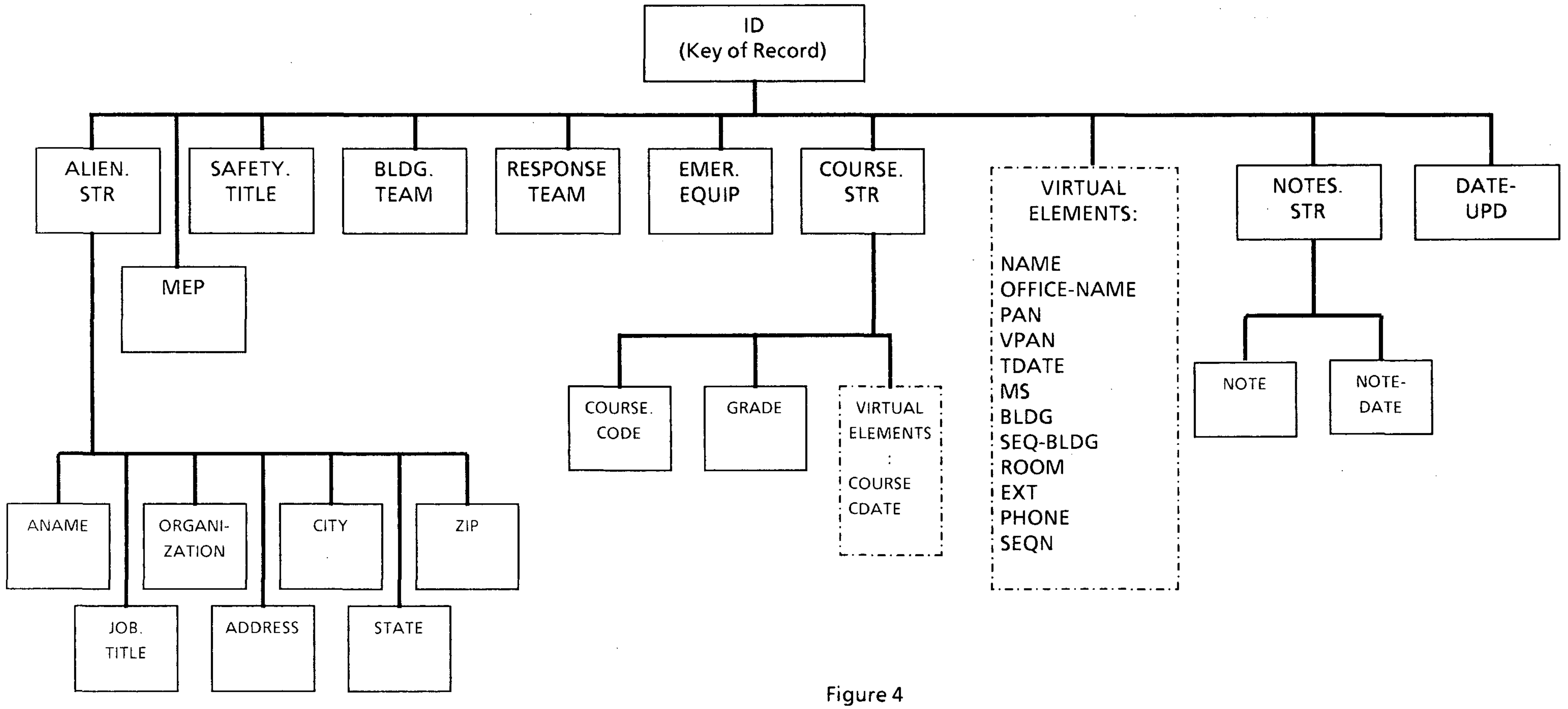

ID
(Key of Record)
ALIEN. STR
MEP
SAFETY. TITLE
BLDG. TEAM
RESPONSE TEAM
EMER. EQUIP
COURSE. STR
VIRTUAL ELEMENTS:
NAME
OFFICE-NAME
PAN
VPAN
TDATE
MS
BLDG
SEQ-BLDG
ROOM
EXT
PHONE
SEQN
NOTES. STR
DATE-UPD
COURSE. CODE
GRADE
VIRTUAL ELEMENTS
:
COURSE
CDATE
NOTE
NOTE-DATE
ANAME
JOB. TITLE
ORGANI-ZATION
ADDRESS
CITY
STATE
ZIP

Figure 4

V.1 Description of elements in the STUDENTS subfile.

Element Name	Required/Opt	Length	Occurrences	Data Type	Indexed
ID (key of the record)	Required	Fixed	Single	String	Immed. indexed
MEP	Optional	Variable	Single	String	
SAFETY.TITLE (ST)	Optional	Variable	Multiple	String	Immed. Indexed
BLDG.TEAM (BT)	Optional	Variable	Multiple	String	Immed. Indexed
RESPONSE.TEAM (RT)	Optional	Variable	Multiple	String	Immed. Indexed
EMER.EQUIPMENT (EQUIP)	Optional	Variable	Multiple	String	Immed. Indexed
NAME	Virtual			Virtual	Phantom Index
OFFICE-NAME	Virtual			Virtual	
PAN	Virtual			Virtual	Phantom Index
VPAN	Virtual			Virtual	for passing
TDATE	Virtual			Virtual	
MS	Virtual			Virtual	Phantom Index
BLDG	Virtual			Virutal	Phantom Index
SEQ-BLDG (SEQB)	Virtual			Virtual	
ROOM	Virtual			Virtual	Phantom Index
EXT	Virtual			Virutal	Phantom Index
PHONE	Virtual			Virtual	
SEQN	Virtual			Virtual	
Alien.Structure	Optional	Variable	Single	Structure	
ANAME	Optional	Variable	Single	Name	Immed. indexed
JOB.TITLE (JT)	Optional	Variable	Single	String	
ORGANIZATION (ORG)	Optional	Variable	Single	String	
ADDRESS (ADD)	Optional	Variable	Multiple	String	
CITY	Optional	Variable	Single	String	
STATE	Optional	Variable	Single	String	
ZIP	Optional	Variable	Single	String	
COURSE.STR	Optional	Variable	Multiple	Structure	
COURSE.CODE (CC)	Required	Variable	Single	Sting	Indexed/ lookup
GRADE	Optional	Variable	Single	String	
COURSE	Virtual			Virtual	Immed. Indexed
CDATE	Virtual			Virtual	
NOTE.STR	Optional	Variable	Multiple	Structure	
NOTE	Optional	Variable	Single	String	
NOTE-DATE	Automatic	Fixed	Single	Date	
DATE.UPD	Required (automatic)	Fixed	Single	Date	

V.2 Searching and Displaying Records. in the STUDENTS subfile

To use the STUDENTS subfile, you must SELect it with the command:

SELect STUDENTS

If you do not know the ID for a record which you wish to see (the student's ID), then you must search for it based upon some criteria you do know. Use the FINd command to search for records in this way Then, to look at the records which are the result of a **FINd** command, enter the command **TYPe.** All of the records in the search result will then be displayed.

You may search for a student's records based on any element or combination of elements. However, some elements are used as the basis of searching much more often than others. Those elements are *indexed* in the same way as explained in section IV.3.

To see a list of the elements in the subfile, enter the command **SHOw ELEMents.**

To see a list of indexes, enter the command **SHOw INDexes.**

As indicated in the element list above, those elements which are indexed are:

ID (key of the record)
SAFETY.TITLE
BLDG.TEAM
RESPONSE.TEAM
EMER.EQUIP
NAME
PAN
MS
BLDG
ROOM
EXT
ANAME (with name)
COURSE.CODE
COURSE

Use the FIND command, as follows:

FIND <index name> <relational operator> <value>

For example, to find all the students that have taken first aid, enter:

FIND COURSE = FIRST AID

Then use the TYPe command to show the result.

If you do not include the relational operator in your search, SPIRES assumes an "equals" operator:

FIND COURSE FIRST AID

Less commonly-used elements are not indexed, for example, COURSE.DATE. To search for all the courses with a specific course date, enter

FOR TREe WHEre <non-virtual element> = <value>

For example,

FOR TREe WHEre ORG = RED CROSS

Then use the DISplay <first/all/last/n/etc> command to show the result.

NOTE: GLOBAL FOR does not work with virtual elements in this version.

Immediate Indexing. Indexes are now updated immediately when a record containing indexed elements is updated. It is not necessary to wait for overnight processing for the indexes to reflect record updates. The FINd command will find a record immediately after an UPDate command is issued.

Goal-as-index Searching. Goal-as-index searching is implement allowing the user to use the find command on the key as well as other elements that are indexed:

FINd <COURSE.CODE>
TYPe

Indirect Searching. A major improvement in searching phantom elements is "indirect searching". Rather than building indexs from data that does not exist (e.g., building indexes on employee information such as EXT and MAILSTOP), it is now possible to use the indexes in the LBLSTAFF file to retrieve records in the TRAINGING subfile. This is transparent to the user. For example to find all students in BLDG 50, issue the find command:

FINd BLDG 50

SPIRES then searches the LBLSTAFF building index, and obtains an intermediate result. SPIRES the compares the records in the intermediate result with those student records in the TRAINING subfile that have the same keys, and reports this final result to the user. Thus, if the LBLSTAFF building index had 300 records with BLDG = 50, and of those 100 student record keys matched 100 of the 300 LBLSTAFF record keys, SPIRES will report a result of 100 student records.

Because the NAME element searches an index in LBLSTAFF, ALIEN names are not longer searchable at the same time. A new index on ANAME is available.

A complete description of all the searching capabilities in SPIRES is described in the document Searching and Updating listed in Appendix E.

V.3 Updating records.

Use the TRAnsfer and UPDate procedure as described in Sect. IV.4 above. **Values for COURSE.CODE must exist in the COURSES subfile before they can be used in a STUDENTS record. If an error message indicates that the COURSE.CODE element value is invalid, xedit ACTIVE FILE to check for typograpical error. If you believe the value to be correct, SELect the COURSES subfile to check the entry (Sections IV).**

When adding or modifying data elements, remember that the format is:

```
DATA ELEMENT  =  value  ;
```

Don't forget the semicolon!

Further, if adding a note, insert the following lines:

```
NOTE.STR;
NOTE = <text of note, no limit to length> ;
```

If adding a course, insert the following lines:

```
COURSE.STR;
CC = <course.code of the course to be added> ;
GRADE = <grade>;   (optional)
```

Don't forget the semicolons!

Indexes are now updated immediately when a record containing indexed elements is updated. It is not necessary to wait for overnight processing for the indexes to reflect record updates. The FINd command will find a record immediately after an UPDate command is issued.

V.4 Adding records.

First, determine whether the student's record already exists. To do so, use the DISplay command:

```
DIS <employee-ID>
```

If the record *already exists*, examine the data to be added to determine that the course.codes are valid and exist in the COURSES subfile. If not, follow the instructions in Section IV to add them. Then TRAnsfer, edit, and UPDate the student's record to add the new course to the student record (Sect V.3)

If the record *does not exist* in the STUDENTS subfile, examine the data to be added to determine that the course.codes are valid and exist in the COURSES subfile. If not, follow the instructions in Section IV to add them. Once you are sure the values to be added are valid, then to add a new STUDENTS record:

SEL STUDENTS (if not already SELected)

SET FORMAT $PROMPT

ADD

You will then be prompted for the value of each element. Here's a sample session showing how to add a record (system responses in **bold**):

```
sel STUDENTS   [CR]
-?
set format $prompt   [CR]
-?
ADD   [CR]
:ID
e999999    [CR]
   Struc:   ALIEN
:       ANAME
[CR]
   Struc:   COURSE.STR
:       COURSE.CODE
ehs0010.8110  [CR]
:       GRADE
[CR]
   Struc:   COURSE.STR
:       COURSE.CODE
[CR]
:  SAFETY.TITLE(1)
[CR]
:  BLDG.TEAM(1)
[CR]
:  RESPONSE.TEAM(1)
[CR]
:  EMER.EQUIPMENT(1)
[CR]
   Struc:   NOTES.STR
:       NOTE
[CR]
-?
dis e999999
ID = E999999;
COURSE.STR;
   COURSE.CODE = EHS0010.8110;
   COURSE = SAFETY ORIENT;
   CDATE = 10/02/81;
NAME = SAM STUDENT;
PAN = 9191;
MS = 50B-2258;
BLDG = 50B;
SEQ-BLDG = "  50B";
ROOM = 2258B;
EXT = 5458;
SEQN = STUDENT, SAM A.;
```

V.5 Removing Records.

A complete description of REMoving records in SPIRES is described in the document Searching and Updating listed in Appendix E. Below, a very brief summary and sample session provides an adequate overview.

To REMove a record in the STUDENTS subfile, enter the command:

REM <ID>

For example, to REMove record E999999, enter:

REM E999999

Because the TRAINING system has multiple users, the system must prevent one user from removing a record that another user is still using. Hence, if you are not using the owner virtual machine, REMove commands may fail with an error message indicating that you do not have the privilege to remove the data elements belonging to another user. If the REMove fails, enter a new occurrence of the notes structure which mentions that the record should be removed and why. Also include your name and extention (Section V.3).

Note: If you wish to restore a record that was erroneously REMoved, please call for assistance. The DEQueue command may be disabled in this subfile.

If you have verified that no records exist in the STUDENT subfile which refer to the course to be removed, and you still received **PRIVILEGED COMMAND**, then TRAnsfer and Xedit the record to add an occurrence of the NOTES element In the note, indicate that the record should be removed any why, along with your name and extension. The database owner will periodically scan these to accomplish the removal.

V.6 Changing the Key of a STUDENTS record.

The key of a STUDENTS record (ID) **cannot** be modified by editing its value and the issuing an **UPDate** command. To change the key of a record, you must:

1. TRAnsfer the old record into your ACTIVE FILE
2. edit the ID to the new number
3. ADD
4. **DIS**play < *new* ID> (to verify that the record looks correct.
5. **REM**ove <old ID> (as described in Sect. V.4 above.))

NOTE: If the REMove fails, as explained in V.5, then there will be two records for the same student, the old record and the new one. You should immediately call your database administrator to have all the information for a single student gathered in a single record.

Please ask for human help if you encounter any difficulties (Appendix F).

VI. Using the SERVICE subfile

VI.1 Description of elements in the SERIVCE subfile.

The SERVICE subfile is a subset of the LBLSTAFF database. SERVICE is used primarily by the telephone operators and mailroom personnel for online realtime retrieval of employee telephone extentions and mailstops. The data is maintained by the Telephone Services Department. The SERVICE subfile default format appears:

```
EMPLOYEE-NAME        BUILDING   EXTENTIONS           DATE LAST CHANGED
MAILSTOP             PAYROLL ACCOUNT NUMBER          DIVISION
```

VI.2 Searching in the SERVICE subfile

To use the SERVICE subfile, enter

SELect SERVICE

Normal SPIRES searching commands are unnecessary in the SERVICE subfile when searching for employee names. When the subfile is SELected, the system responds with the prompt:

```
ENTER SEARCH STRING:
```

Simply enter a surname alone, the first part of a surname, or all or part of the given name and all or part of a surname.

E.g., to find Ernest O. Lawrence, any of the following search strings are valid:

```
LAWRENCE
LAWRE
LAW
E LAWRENCE
E LAWREN
E LAW
E O LAWRENCE
E O LAW
O LAWREN
ERN O LAW
ERNEST O LAWRENCE
```

To exit the prompting routing, enter an asterisk:

```
ENTER SEARCH STRING: *
```

When in the SERVICE subfile and exited from the automatic searching facility ("ENTER SEARCH STRING"), you may use normal SPIRES search commands such as SHOw ELEMents, SHOw INDexes, FINd, TYPe, and DISplay.

To turn the automatic prompting back on for name searching, SELect SERVICE.

VI.3 Updating, adding, deleting records.

Updating, adding and deleting records in the SERVICE subfile is prohibited.

VII. Procedure for entering data for a New Class

When a new occurrence of a course (i.e., a new class) is taught, it is probable that a roster will be generated, which should include the following information:

Course data:	Course title, e.g., first aid Course number Hours Sponsor Name Course date Instructor
Student data	Student name Employee-ID (or guest ID), if any If not an employee (or guest), phone number, address, etc.

Given a sheet with such information, it is desireable that all of the pertinent data be entered and verified easily. There are three approaches from which we must choose:

A. We could enter student ID into a course record, and require that the system validate the student ID. However, if the student incorrectly entered his ID number on the class roster, there is a chance that the ID would still be valid but for the wrong student. Also, if the student's ID has changed since his record was entered, the new pointer would not point to the existing record. Thus, though this saves time by entering all the data in the COURSES record, it still requires looking at the student record to verify that the correct student was affiliated with the class. But worst, if an employee number changed for a given employee, one would have to go in to each course record for each class in which that individual participated and change his employee number.

B. We could enter the course code in the student record and require that the system validate the course code. This insures that we will look closely enough at the student record to insure that it is the correct one. This is also the best way to detect when an employee's ID number has changed. Also, there is much less chance of an erroneous course code pointing to a valid but incorrect class. However, it is time-consuming to update each employee's record individually.

C. We could create an interface in the form of an exec that would prompt for each course element, then add the course record, then prompt for each student ID, use that ID to display the records in both STUDENTS (student subfile) and LBLSTAFF for comparison, and, if approved, enter either the student ID in the course record, or the course ID in the student record. This provides all required validation while economizing on data entry time. Entering the student ID's in the COURSES record would be preferable. It boils down to a choice between:

(A) multiple student entries into 1 course record
(B) multiple course entries into 1 student record

Approach C makes the dilema transparent as it provides all the validation and ease of use. It can be implemented as time becomes available.

At the present time, Approach B is used to insure accuracy since the need for correcting course ID's is much less frequent than changing employee ID's.

The procedure (for Approach B) is as follows:

1. Add a new record to the COURSES subfile to reflect the information for a new occurrence of the course.

2. SELect SERVICE, and search each name on the roster to verify the student ID. This can be done quite rapidly.

   Indicate on the roster those ID's which do not match and those students which do not occur in SERVICE at all. If the student's record in the SERVICE subfile cannot be found using their name, try DISplaying the ID which they provided directly. To do so, use the asterisk as discussed in Sect III.2. For those students which appear to be LBL employees but for whom the ID number cannot be verified in SERVICE, contact the student or the Personnel Dept.

3. Now begin adding course data to students records. SEL STUDENTS. Then, for each student whose ID was validated in step 2 above:

   . DIS <ID>

     If the record exists, TRAnsfer, edit and UPD as described in Sect IV.4

     If the record does not exist, ADD it as described in Sect IV.5

4. It is usually a good idea to then SELect COURSES and DISplay the course (Sect IV.) If you wait until the following day, it will DISplay the course along with all of the students which you entered in that course through the previous day.

VIII. Generating Reports.

Occasionally you may wish to use some other format than the default SPIRES format. Customized formats have been provided, as described in section. The command SHOw FORmats will list these and also indicate if either one is currently in effect by notating " - SET" after the format name.

RULE: If any virtual element is **SET**, then the format will not take effect. That is, presence of a virtual element overrides a format. Thus, in order to use a format, you should enter the command **CLR ELEM** beforehand. The same restriction applies to generating tables with the **SET FORmat $REPORT** formats. Virtual elements must be cleared by **CLR ELEM** before the report will take effect. The report-generating EXECs listed below perform this automatically.

To enable staff to produce these reports easily, a SPIRES protocol is provided to produce and print pre-defined reports.

As mentioned in Section II, nearly any CMS file that the user has created can be printed using the LPR and LPRCC commands. The REPORT exec includes the printing options so that the user does not have to issue print commands separately. However, the reports are stored as CMS files and can be reprinted at will using the LPRCC command (the reports all use column one for carriage control).

NOTE: Before generating a report, it may be useful to validate the data in the database. To do this:

```
SEL STUDENTS
CLR ELEM
DEFine TABle ID NAME MS TDATE
FOR SUBFILE
IN ACT CLR DIS ALL
```

You may then see the results: **(X ACTIVE FILE)**
or print the file: **(LPRCC ACTIVE FILE)**

NOTE: The output will be printed on the Talaris laser printer on the first floor of Bldg 50B and filed in the slot for BITNET. **The best time to generate a report is before 10 a.m.**

The following EXEC produce reports from the TRAINING and COURSES subfiles. They are initiated by typing the name of the EXEC followed by a carriage return.

Report Name	Report produced
BLDG	description to be provided
BLDGEMRG	description to be provided
BLDGRPT	description to be provided
BUILDING	description to be provided
CLASS	description to be provided
COURSE	description to be provided
JOYCE	Adhesive mailing lables
MAKLABEL	description to be provided
PAN2	description to be provided
PHYSICS	description to be provided
RHODES	description to be provided
ROB	description to be provided
SPEC	description to be provided

TEAM	description to be provided
TEAM2	description to be provided
VALDATE	Validates data in the database, see above.

IX Printing; the LPR, LPRCC, and LABEL commands

Staff may occasionally wish to print files other than standard reports. These may include a file created using the Xedit editor, or created by SPIRES as the result of a FIND or DISPLAY command. (SPIRES usually places search result displays and other output in the CMS file named ACTIVE FILE A or on the CRT or both.) Therefore, it will often be useful to be able to print files directly. There are two EXECs that will send files to the IBM 3203 printer with dual-size paper on the first floor of Bldg. 50B in the Central Computing Facility machine room area. These EXECs are:

LPR and **LPRCC**

The syntax of these commands is:

LPR <filename> <filetype> <filemode>

LPRCC <filename> <filetype> <filemode>

For example, to print the CMS file, ACTIVE FILE A, enter the command:

LPR ACTIVE FILE A

The distinction between the two is that **LPRCC** interprets any characters in the first column of the file (at the left margin) as carriage control (hence the CC; LPR is an acronym for line printer). Generally, users will not insert carriage control characters in a file, and so **LPR** is the appropriate command to use. However, the SPIRES facility **DEFINE TABLE** and **FORMAT $REPORT** automatically reserve column 1 for carriage control characters, with data beginning in column 2. For files generated by these utilities, **LPRCC** should be used.

A file probably includes carriage control if most of the text begins in column two and column one contains characters such as: 1, 0, and +. For example, it may look something like:

```
1
 MARY HAD A LITTLE LAMB
 ITS FLEECE WAS WHITE AS SNOW
+               WHITE AS SNOW
0
 AND EVERY WHERE THAT MARY WENT
 THE LAMB WAS SURE TO GO.
```

The LABEL command is used to print an existing file in label format. label format is :

Column one blank except for a "1" on the first line.
No more than 40 characters per line.
No more than eight lines per label.

It is advisable to check the label printer prior to sending a file to the label printer. So send a file do:

```
LABEL filename filetype filemode
```

E.g.,

```
LABEL STUDENT LABEL A
```

Both label and laser printers are on the first floor of 50B in the User Area.

APPENDIX A

DIP-SWITCH SETTINGS FOR ADM-3A TERMINALS FOR USE ON SERIES/I 3270 EMULATOR

INTERNAL

	ON	OFF		ON	OFF
none			7	X	
6	X		6		X
5		X	5	X	
4	X		4		X
3		X	3		X
2		X	2		X
1		X	1		X

EXTERNAL

	ON	OFF
Bit 8-0	X	
Parity	X	
STOP	X	
Data 7	X	
Parity		X
LC	X	
Auto NL	X	
RS232	X	
HDX		X
All speeds but 9600		X
9600	X	

APPENDIX B

TERMINAL CONTROL

The SERIES/1 terminal controller commands are summarized in the document "Key Definitions for IBM 3277 Terminal Emulation", section, "ADM-3A Key Definitions for IBM 3277 Terminal Emulation" available from the Electronics Shop in 50B-2259 (see Saul Duenas). Other ASCII terminals such as the VT100 may be used as well. Each has its own key definitions which are summarized in the same document.

Occasionally, the system will not accept characters typed on the keyboard, but rather sound the "bell". To clear this keyboard lock, depress the CONTROL key and, while depressed, enter the letter sequence: RTXQV. This is notated

CNTL-RTXQV

When the system is displaying output on the CRT screen, it will stop after 22 or 23 lines, depending on the kind of terminal. The message **MORE** wil be displayed at the lower right. At this point, one has four options:

1. Do nothing. After 50 seconds, the bell will sound. After an additional 10 seconds, the system will clear the screen and display the next page.
2. Enter **CNTL-Z**. This causes the next 23 lines to be displayed immediately.
3. Enter a [CR]. This causes the message in the lower right portion of the screen to change from **MORE** to **HOLDING**. The timer holds, and the screen will not change. Another [CR] causes the message in the lower right to return to **MORE** and the timer is reset.
4. Enter **HT** [CR], then **CNTL-Z**. The **HT** halts typing, preventing the rest of the lines from being displayed. The **CNTL-Z** then clears the screen .

Several helpful CMS terminal commands are available:

The pound sign (#) acts as a LINEND character (line end).
The double-quote (") acts as an ESCAPE character
The (@) acts as a CHARDEL (character delete) character.
The (¢) acts as a LINEDEL character (line delete)

The (#) and the (") have been disabled as CMS control characters since they confilict with often-used SPIRES characters.

<u>Series/1 - ADM3A terminal control commands</u> (The complete list can be found in "ADM-3A Key Definitions for IBM 3277 Terminal Emulation" available from the Computer Center Library.):

CNTL-N	go to next line
CNTL-H (or left-arrow key)	move cursor to the left
CNTL-L (or right-arrow key)	move cursor to the right
CNTL-K (or up-arrow key)	move cursor up
CNTL-J (or down-arrow key)	move cursor down
CNTL-D	deletes a character
CNTL-E	deletes a line
ESC-spacebar	enter or leave *character* insert mode

These sequences work in the editor as well as outside the editor.

Program Function (PF) keys

In some utilities, such as FLIST and Xedit, PF keys are assigned specific functions. When using an ADM-3A terminal, the PF keys are implemented as a sequence of two keys: the **ESC** key followed by some other key. For PF1 through PF9, use **ESC 1** through **ESC 9**. **ESC-:** (colon) is equivalent to **EXC-11**, and deletes to the end of line in the FLIST facility. **ESC-3** usually means "quit". **ESC-1** usually calls a CMS help screen. Often a menu of valid PF keys will be displayed in utilities where they are recognized.

APPENDIX C

SPIRES FORMAT $PROMPT Subcommands

The following commands are recognized by SPIRES when adding new records (or modifying existing records) using SET FORMAT $PROMPT (formerly SET INPUT FORMAT):

[CR] (carriage return)	Continue to next prompt
//	Puts in a null-length value if legal, otherwise you are reprompted for a legal value.
/N	Skip to the next element of the current structure for input
/S	Skip to the next structure for input (first element of next structure)
/ <value>	Retains leading blanks (blanks in front of the value)
<value> //	Continue value on next line (for long values, e.g., paragraphs)
/E	End input for the current structure, and retain input thus far
/X	Abort input, and do not retain any input

Example of //: to enter a null value in a structure without exiting the structure, for example in the ALIEN structure:

```
STRUCTURE   ALIEN

          ANAME: //
          JOB.TITLE  president
          (other elements)
```

This prevents the other elements in the ALIEN structure from being skipped merely because there was no value entered for ANAME.

The full set of subcommands can be found in the SPIRES manual Searching and Updating.

APPENDIX D

Looking at your CMS files

The CMS FLIST facility provides a listing of your permanent files and several capabilities to browse, edit, copy, rename, and delete them. To use the FLIST facility, enter the command **FLIST** and your files will be displayed, with the cursor at the top of the list. You may move the cursor up and down to select any file You may use the **ESC** commands on the menu at the bottom to perform various operations, e.g., **ESC-4** or an **X** will invoke the editor on the selected file, an **EXC-2** will allow you to browse the file, and **ESC-8** will allow you to see the next screenful of files on your list if you have more files than can be listed on one screen, and **ESC-3** will exit FLIST. All the terminal control keys work in FLIST.

There are other file listing facilities besides FLIST. FLIST currently provides the most functionality. For assistance with FLIST, please see Appendix F for human help.

APPENDIX E

Documentation

A complete set of SPIRES documentation is available from the Computer Center library. The following are most likely to be of interest to users of the Training database system.:

1. A Guide to Searching -- A SPIRES Primer.
2. Searching and Updating.
3. Sequential Record Processing: Global FOR Reference Manual.
4. SPIRES Keyterm Index -- An index of all SPIRES terms.

A complete set of CMS documentation is available from the Computer Center library. The following are most likely to be of interest to users of the AWARDS database system.:

1. System Product Editor User's Guide (SC24-5220-1)
2. System Product Editor Command and Macro Reference (SC24-5221-1)

The RTSG Electronics Shop in Bldg. 50B-2259 has copies of the following documents:

1. ADM-3A Key Definitions for IBM 3277 Terminal Emulation
2. VT100 Key Definitions for IBM 3277 Terminal Emulation

APPENDIX F

Human Help

For assistance, call:

Clay Sealy	x 7151 or 5831
Joyce Putnam	x 4013
Allan Konrad	x 5458

APPENDIX G

Using Xedit

The following describes use of Xedit with an ADM-3A terminal. For other terminals, please see Appendix B.

(**Note:** If you are using the Xedit editor and SPIRES, be aware that it is helpful to be in the same case mode in the editor as in SPIRES. That is, it is possible to be in SPIRES in upper-and-lower case, while in Xedit in upper only, or vice-versa. The default for the TRAINING system is to be in upper and lower case both in the editor and in SPIRES. If you have problems with case, call for human help (Appendix F).)

Files in the VM/CMS system have three-part names:

filename filetype filemode

usually abbreviated

fn ft fm

The filemode is generally assumed to be A, refering to you "A-disk", 191. This 191 A disk is your private disk.

To edit a file, issue the command

X fn ft

For example, to edit the CMS file ACTIVE FILE A, enter

X ACTIVE FILE A

The document will then appear ready to edit. Case is not significant on this command. You could also enter:

x active file a

If the file ACTIVE FILE did not exist on your A disk, the editor would create a new empty file, with only a top-of-file and a bottom-of-file marker.

Once in the editor, you can:

Use the **CNTL-D** and **CNTL-E** keys (see Appendix B)

Use the "cursor" keys to move the cursor around on the screen. On an ADM3A terminal, depress the **CONTROL** key, and while holding it, press either **H**, **J**, **K**, or **L** depending on which direction you wish to move the cursor. After you release **CONTROL**, whatever characters you type will replace the text in your file, if any.

Use the prefix field on the left side of the screen (the five columns of equal signs) to copy, delete or move whole lines or groups of lines.

Often-used Prefix-field Commands.

D (delete)

To delete one line, place a **d** anywhere in the prefix field to the left of the line you wish to delete. Then hit [CR]. E.g.,

```
===== This is line one
==d== This is line two
===== This is line three
```

results in:

```
===== This is line one
===== This is line three
```

To delete a known number of contiguous lines, enter **d** and the number of lines to be deleted.

```
===== This is line one
==d2= This is line two
===== This is line three
===== This is line four
```

results in:

```
===== This is line one
===== This is line four
```

To delete an *un*known number of contiguous lines, that is, a "block" of lines enter **dd** on the first line to be deleted and on the last line to be deleted. E.g.,

```
===== This is line one
==dd= This is line two
===== This is line three
dd=== This is line four
===== This is line five
```

results in:

```
===== This is line one
===== This is line five
```

I (insert)

To insert a new blank line that can be edited, place an i in the prefix field on the line which you want the new line **to follow**. E.g,

```
===== This is line one
==i== This is line two
===== This is line three
===== This is line four
```

results in:

```
===== This is line one
===== This is line two
=====
===== This is line three
===== This is line four
```

The new blank line can now be edited by moving the cursor to anywhere to the right of the prefix field and the first blank column following it.

To insert a specified number of new blank lines that can be edited, place an **i** and the number of blank lines needed in the prefix field on the line which you want the new line **to follow**. E.g,

```
===== This is line one
==i3= This is line two
===== This is line three
===== This is line four
```

results in:

```
===== This is line one
===== This is line two
=====
=====
=====
===== This is line three
===== This is line four
```

It is also possible to insert lines by entering the command i on the command line at the bottom of the screen. This will clear the screen below the column-counter line. You can then enter text and use **CNTL-N** to go to the next line. When you hit a [CR], your text will be shifted up above the column-counter line and the lower part of the screen will be available for more input. Two consecutive [CR]'s will return you to normal edit mode.

C (copy)

To copy one line, place a **c** anywhere in the prefix field to the left of the line you wish to copy and a **p** on the line before which the newly created line should be placed. E.g,

```
===== This is line one
==c== This is line two
====p This is line three
```

results in:

```
===== This is line one
===== This is line two
===== This is line two
===== This is line three
```

the **p** stands for *prior* and instructs the system to put the new copy of the line prior to the line with the **p**. You can use the **f** instead, which means *following:*

```
===== This is line one
==c== This is line two
===== This is line three
===f= This is line four
```

results in:

```
===== This is line one
===== This is line two
===== This is line three
===== This is line four
===== This is line two
```

To copy a known number of contiguous lines, enter **c** and the number of lines to be copied on the first line to be copied, and an **f** or a **p** to mark where the copied lines should be placed:

```
===== This is line one
==c2= This is line two
===== This is line three
===f= This is line four
```

results in:

```
===== This is line one
===== This is line two
===== This is line three
===== This is line four
===== This is line two
===== This is line three
```

To copy a *un*known number of contiguous lines, that is, a "block" of lines, enter **cc** on the first line to be copied and on the last line to be copied, and an **f** or a **p** to mark where the copies should be placed:

```
==p== This is line one
==cc= This is line two
===== This is line three
cc=== This is line four
===== This is line five
```

results in:

```
===== This is line two
===== This is line three
===== This is line four
===== This is line one
===== This is line two
===== This is line three
===== This is line four
===== This is line five
```

M (move)

the move command, **m**, works similarly to copy:

```
===== This is line one
==m== This is line two
====f This is line three
```

results in:

```
===== This is line one
===== This is line three
===== This is line two
```

and,

```
=p=== This is line one
==mm= This is line two
===== This is line three
===mm This is line four
```

results in:

```
===== This is line two
===== This is line three
===== This is line four
===== This is line one
```

Most terminals can only display about 22 lines of text. Therefore, if the file you are editing is longer than 22 lines, not all of them can be displayed simultaneously.

Think of your file as if it were a very tall building. The building is a strange building however, because its floors are numbered from top to bottom rather than from bottom to top! So the first floor is at the top of the building.

Our building has a rather unique elevator. Unquestionably the oddest thing of all is that the elevator doesn't move, the building does! The elevator is fixed, but the building moves up and down, into and out of the ground.

But that's not all! First, its doors are always open, so you can always see out as the building moves up and down in front of you. Furthermore, your elevator is 21 stories high! Stranger yet is that half-way up this tall elevator is a platform on which you stand. Thus, you can see the floor that is level with yourself, the 10 floors lower, and the 10 floors higher.

This peculiar building is like your file and your terminal is like its elevator which provides you with a view of some portion of of the building. Imagine standing in the fixed elevator as the building moves up and down in front of you. This is exactly the phenomenon you experience using the editor.

When you first enter the editor, it automatically gives you a view of the top 10 lines of your file. This is like standing in your elevator at the top of the building, with a view of the 10 floors beneath you and 10 stories of thin air above you.

If you wish to look at lower floors of the building, what would you do? You would command the building to shift **up** (which is equivalent to the elevator going down). This is exactly what you do in the editor. The following is a brief summary of the commands that you can use to move around in your file. They are entered on the command line at the bottom of your screen when you're in the editor.

+5 shifts the file up 5 lines so that your view is the next 5 lines **down.** The "+" is optional. Just a 5 or any number is acceptable.

To adjust your view in the opposite direction, i.e., towards the top of the file, use a minus sign preceding the number of lines you want to shift, e.g., **-20** will display the portion of the file 20 lines above your current position.

The command **top** will go the the top of the file. The command **bot** will go to the bottom of the file.

When a number is preceeded with a colon, the editor will go directly to that absolute line number. E.g, **:104** would display lines 93 through 115, with line 104 exactly in the middle of the screen.

To locate a string of characters, enter a slash (/) and the character string to be searched for. It will locate the first instance of that string. If you want to search for later occurrences, continue entering equal signs (=) until you find the occurrence you desire.

Finally, the insert command, **i**, discussed above, is entered from the command line and allows you to insert a virtually infinite number of new lines at that point in the file.

It would not be useful to give every detail of the editor here. See Appendix E for a list of documents which describe how to use the editor. If you need assistance, please see Appendix F for human help.

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720