The views, conclusions, or recommendations expressed in this document do not neces-sarily reflect the official views or policies of agencies of the United States Government.
This document was produced by SDC and III in performance of contract AF 19(628)-5166 with the Electronic Systems Division, Air Force Systems Command, in performance of ARPA Order 773 for the Advanced Research Projects Agency Information Processing Techniques Office, and Subcontract 65-107.

TECH MEMO
a working paper

System Development Corporation / 2500 Colorado Avenue / Santa Monica, California 90406
Information International Inc. / 200 Sixth Street / Cambridge, Massachusetts 02142

TM- 2337/100/01
AUTHOR S. L. Kameny
L. Hawkinson
L. Hawkinson
TECHNICAL

RELEASE S. L. Kameny, SDC
E. Fredkin
E. Fredkin, III
for J. I. Schwartz
DATE 4/13/66 PAGE 1 OF 19 PAGES

LISP Edit Program LISPED

ABSTRACT

This document describes the LISPED program, which is a context editor for LISP data and exists as a separate program under time-sharing.

INTRODUCTION

LISPED is a context editor for LISP data. It exists as a separate version of LISP containing master function LISPEDIT together with its subsidiary function STRINGED and approximately fifty additional subsidiary functions used by LISPEDIT and STRINGED.

LISPEDIT was written by Lowell Hawkinson of Information International, Incorporated, to facilitate the LISP 2 effort, on which III is working under subcontract to System Development Corporation.

CONTENTS

Tables

1. LISPED MODES

LISPED operates in three modes. LISPEDIT and Evalquote modes are described in the remainder of Section 1. STRINGED is described in Section 2. Within the descriptions, the following notations will be utilized.

f means LISP library file name. A file name is any LISP literal atom. It serves no other purpose than to identify a particular file.

ℓ means either a single file name or a list of file names enclosed in parentheses.

LISPEDIT

This is the normal mode in which LISPED is entered. Any error encountered within LISPEDIT returns to LISPEDIT mode.

Evalquote

The Evalquote mode is similar to the normal mode of operation of Q-32 LISP, operating successive pairs of S-expressions and printing out the results. Evalquote is entered from LISPEDIT by the command EVQ, and continues until either EXIT or LISPEDIT is given as the first S-expression to Evalquote, in which case LISPEDIT mode resumes. An error occurring in Evalquote mode returns to Evalquote mode.

STRINGED

The STRINGED mode is entered from LISPEDIT by means of the commands INPUT, EDIT or STRINGED. The function available in STRINGED mode are described in Section 2. STRINGED mode continues until the EXIT command is accepted by STRINGED (either the FILE command followed by EXIT, or else EXIT EXIT must be used). An error occurring within STRINGED returns to STRINGED mode.

The system is relatively foolproof in that library files are protected from damage in the event of LISP unwind. After an unwind, the system always returns to the same state in LISPEDIT or STRINGEDIT which it was in before the error occurred.

LISPED is, in general, a talkative system and most of the messages printed out have a relatively simple interpretation.

1.1 LISPEDIT

LISPEDIT is the normal mode in which LISPED is entered. If the system is clear, the entrance into LISPEDIT is signified only by a double bell. If the system is re-entered following an error, the statement

```
    LISPEDIT RECOVERY // FILES SAVED
```

is given followed by two bells to indicate that LISPEDIT is ready for input.

The LISPEDIT mode may be used for inputting and editing LISP data, reading LISP data from tape, performing general file maintenance operations, and running and testing of LISP programs.

The LISPEDIT commands are listed in Table 1 and described in Section 1.2.

1.2 LISPEDIT COMMANDS

LISPEDIT accepts the 16 commands given in Table 1. Commands other than OPEN and SHUT are followed by their arguments without parentheses. A final space must always be used except in the case of a list type argument.

OPEN (filename unit optional)

OPEN works in LISPEDIT mode exactly the same way it works in LISP Mod. 2.6 or in Evalquote mode of LISPED. _Filename_ is a literal atom; _unit_ is either DISC (or DISK) or a tape reel number; and _optional_ is either absent or WRITE if a tape file is to be opened for writing or PERM if _filename_ is a previously existing disc file and _unit_ is DISK.

Files opened in either Evalquote mode or LISPEDIT mode can be read in either mode.

In LISPEDIT mode, the response to OPEN is

```
    IO FILE filename OPENED  //  CONTINUE
```

SHUT (filename optional)

SHUT works in LISPEDIT mode exactly the way it works in LISP Mod. 2.6 or in Evalquote of LISPED. _Optional_ is either DELETE for a disc file to be deleted, or else is absent.

Files opened in either LISPEDIT mode or Evalquote mode can be shut in either mode.

In LISPEDIT mode, the response to SHUT is

```
    IO FILE filename SHUT  //  CONTINUE
```

Table 1. LISPEDIT Commands

Name	Arguments
COMBINE	f f f
DELETE	ℓ
EDIT	f
EVQ	-
FILES	-
INPUT	-
LIST	f
OPEN	(filename unit optional)
READ	filename n
REORDER	ℓ
RUN	f
RUNSPEAK	f
SHUT	(filename optional)
STRINGED	f
WRITE	filename n ℓ

READ filename n

Reads the n^{th} file from file filename and adds the contents to the list of current library files. (Note that if filename refers to a disc file, n must be 1. If filename refers to a tape file, then no check is made to see that there are n physical files on the reel. When done, the READ command prints the message

```
    FILES READ FROM filename // CONTINUE
```

WRITE filename n ℓ

If filename refers to a disc file, n must be either 1, to rewrite the file, or 2, to append the current library files onto the end of the previous library files on disc.

If filename refers to a tape file, then WRITE rewinds the tape, skips (n-1) physical files, then writes. Note that no check is made to assure that there are (n-1) physical files on the tape.

After the file is positioned, then if ℓ is any atom, e.g., ALL, all current library files are written onto it.

If $\ell = (f_1\ f_2\ \ldots\ f_n)$, where f_1, f_2 ... f_n are names of library files, then the library files f_1, f_2 ... f_n are written in that order onto the output file filename. After writing the output file, WRITE writes an end-of-file on the output device, and prints out the message

```
    FILES WRITTEN ONTO filename // CONTINUE
```

INPUT

INPUT causes LISPED to go into the STRINGED mode with an initially empty string. The only acceptable STRINGED command which can reasonably be used at this point is INSERT to enter new LISP data into a system. This data will be of use only if eventually a FILE command is given to supply it with a file name.

EDIT f

This is the same as INPUT except that the STRINGED is entered with the string equivalent to the file named f.

STRINGED f

This is identical to the command EDIT f.

RUNSPEAK f

The file named f, which must consist of Evalquote pairs, is operated. Each successive pair of S-expressions in the file f is passed to Evalquote and the results printed.

RUN f

This is the same as RUNSPEAK except that the Evalquote output is not printed.

FILES

This command results in printout of a list of names of all current files.

DELETE ℓ

ℓ can be either a single library file name or a list of library file names. After checking appropriately to see that these names are all names of files in the system and that the user really wanted to delete these files, this command will cause the library file or files to be deleted. In response to question from DELETE, a user should respond with either YES or NO.

REORDER ℓ

If ℓ is a single file name, the named file is placed at the end of the list of current files. If ℓ is a list of file names, these files are removed from the current list of active files and placed in the order named at the end of the list of files.

COMBINE f_1 f_2 f_3

The contents of f_1 and f_2 are concatenated and inserted as a new file named f_3. If a file named f_3 is already in the system, the new file named f_3 will replace it.

LIST f

The command LIST f causes the contents of file f to be printed, one S-expression per line, with a line skipped between S-expressions.

EVQ

The EVQ command causes LISPED to enter the Evalquote mode in which successive pairs of S-expressions input from the teletype are passed to Evalquote and the results printed on the teletype. The Evalquote mode is similar to the normal operating mode of LISP 1.5. However, the LISP SAVE function will save the current version of LISPED in the current state. This will cause a minor inconvenience when the saved version is loaded. The system will automatically go into the EVQ mode and an EXIT will have to be taken to get into LISPEDIT mode.

If the LISPEDIT system is saved using the TSS SAVE function, i.e., !SAVE name or !SAVE, then when the saved version is loaded the LISPED system will print out

```
LISPEDIT RECOVERY RECOVERY // FILES SAVED
```

followed by

```
mode MODE //
```

where mode is the mode in which the system was saved.

EVQ mode continues until EXIT or LISPEDIT is given as a command to Evalquote.

2. STRINGED

STRINGED is a context editing program which is used to update an existing LISP file or to generate a new one.

When STRINGED is entered from its parent program LISPEDIT, the file to be edited (a series of S-expressions, possibly empty) is converted into the equivalent token string. This string may then be examined and operated on through successive commands input on the teletype and interpretively executed by STRINGED. Whenever a desired transformation of the string has been achieved, a command to file the series of S-expressions equivalent to it may be given. Finally, after all desired editing and filing of the string has been completed, STRINGED may be exited and control returned to LISPEDIT.

2.1 TOKEN STRING EQUIVALENT OF AN S-EXPRESSION

The rules for converting an S-expression to its equivalent token string (from which may also be inferred an inverse transformation) are expressed in the table below:

where x is an S-expression
x* is the token string equivalent to x
LP is the atom "left parenthesis"
RP is the atom "right parenthesis"
DT is the atom "dot"

S-expression x	token string x*
NIL	LP RP
non-NIL atom	x
$(a_1\ a_2\ \ldots\ a_n)$	LP $a_1^*\ a_2^*\ \ldots\ a_n^*$ RP
$(a_1\ a_2\ \ldots\ a_{n-1}\ .\ a_n)$	LP $a_1^*\ a_2^*\ \ldots\ a_{n-1}^*$ DT a_n^* RP
e.g., (A (B NIL) . C)	LP A LP B LP RP RP DT C RP

2.2 VISUALIZATION OF THE TOKEN STRING

At any moment during the editing of the token string, there is a particular portion thereof whose boundaries serve as reference points for whatever action may be called for by the next STRINGED command. This portion of the string is termed the object fragment (or simply the fragment, abbreviated as FR). The object fragment may be empty, or it may include the entire string (which also could be empty).

The token string with its object fragment sub-string should be visualized as in Fig. 2.1. The symbol "t" denotes here an arbitrary token (parenthesis, dot, or other LISP atom). Definitions for left boundary (LB), right boundary (RB), head, and tail are self-evident from the diagram. The directions backwards and forwards mean "towards the head" and "towards the tail," respectively. These six terms will be used extensively in the descriptions of STRINGED commands (Section 2.5).

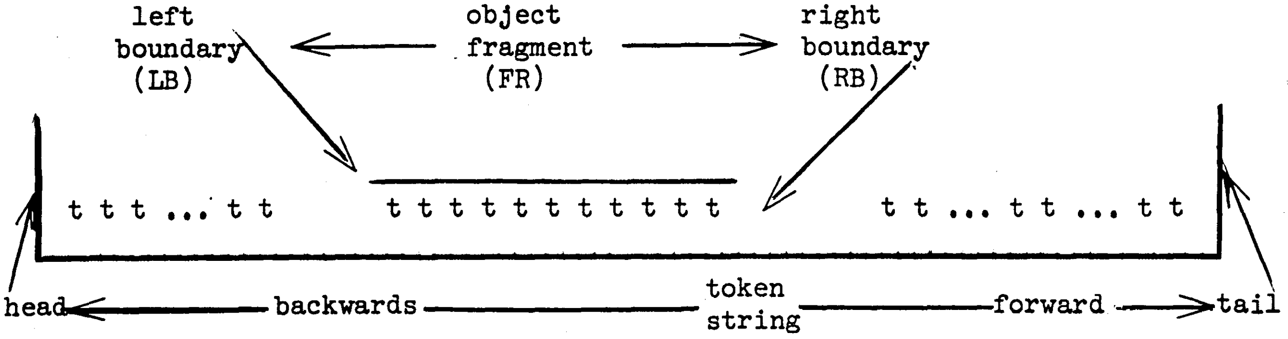

Fig. 2.1 Token String, Object Fragment

2.3 FORMAT OF COMMANDS

Commands to STRINGED are input, one after another, on the teletype. As many commands as fit may be entered on a single line; conversely, any command which would overflow a single line may be continued onto as many additional lines as are necessary.

Each command consists of a name followed by a specific number (zero, one, or two) of arguments. When input, names and arguments must always be followed by one or more blanks, even when they occur as final items in a line. (The algorithm for making the input line-to-line transition, to be followed reflexively, is: type blank and carriage return; allow any output responses to be printed; await double bell signal.)

The name of a STRINGED command is a LISP identifier. All but two of the commands (exceptions are FILE and EXIT) have two interchangeable names: the first is descriptive of the action of the command, the second is a one-(or two) character abbreviation.

Each argument of a command is of one of three types:

n integer
c label
s fragment

An integer argument is simply a LISP fixed-point number. A label argument is any LISP atom other than / or $. A fragment argument has several possible formats, which are discussed in detail in the next section. Should a particular command be given an argument of the wrong type, an appropriate diagnostic will be printed and the remainder of the input line will be ignored.

2.4 FRAGMENT ARGUMENTS

A fragment argument of a STRINGED command has four distinct formats, each of which specifies some particular fragment (sequence of tokens) as its value. These formats, paired with their corresponding values, are listed in the table below. Symbol t denotes an arbitrary token (parenthesis, dot, or other LISP atom). Symbol f denotes the name of a LISP file.

format of s	value of s
/ /	empty
/ $ f	copy of file f
/ c	saved fragment under label c
t t ... t t /	**non-empty input fragment** **t t ... t t**

In the input of a fragment argument, two syntactic rules involving the delimiter blank must be observed.

(1) / and $ must always be separated by one or more blanks

(2) two adjacent tokens (t t) must be separated by one or more blanks when neither is a parenthesis

A saved fragment (third format above) is one which, by some previous command, had been copied from the then current FR and saved for future use under a label c. The saving of FR may be called for explicitly by means of the LABEL command (see its description in Section 2.5).

Should a fragment be saved under a label which is already in use, the new value will supersede (and in fact replace) the old. Thus, only the most recently input and deleted fragments are available at any given moment under a given label.

There **are** three particular tokens (namely /, $/, and ***) which are used as delimiters within an input fragment (fourth format above). All other tokens (parenthesis, dot, or other legitimate LISP atom) are taken literally.

(1) / is the terminal delimiter of an input fragment and must be preceded by at least one literal token.

(2) $/ is an escape delimiter. If it is not the first token in an input fragment, and is not immediately preceded by ***, the command preceding $/ will not be executed, a diagnostic will be printed and the remainder of the input line will be ignored.

(3) *** indicates that the token immediately following it is to be taken literally, even if it would normally be interpreted as a delimiter, e.g., token / is written *** /

2.5 STRINGED COMMANDS

The commands available within STRINGED are listed in Table 2, and are described below.

The command descriptions given below are presented in a standard format. The heading displays, in order, the name of the command, its abbreviation, and an abbreviated specification of the arguments (where n, c, s, and a dash denote integer, label, string, and no arguments, respectively). The first paragraph describes the action of the command when conditions are such that it may be properly carried out. The second paragraph (if any) indicates the conditions required for execution of the command, and what will happen if they are not fulfilled. The assumption is made in the descriptions that all arguments have been correctly entered; where this is not the case a diagnostic will be printed, execution of the command will not be attempted, and the remainder of the input line will be ignored (see Sections 2.3 and 2.4).

The reader is referred to the diagram and definitions in Section 2.2, since all commands are described in terms of them.

EXIT - -

STRINGED is exited and control returned to the supervisory program LISPEDIT. A message confirming successful exit is printed.

If, however, this command was not immediately preceded by a successful FILE command or unsuccessful EXIT command, STRINGED will not be exited, a diagnostic message will be printed, and FR will remain unchanged. This interlock feature prevents a premature, accidental exit from STRINGED. To leave STRINGED without having filed the token string, two EXIT commands must be issued.

FILE - f

The complete token string is first converted into the particular series of S-expressions equivalent to it and then is filed under library file name f for future reference by LISPEDIT commands. If a LISP library file named f already exists, it will be replaced by the newly generated one. A message confirming successful filing is printed, and FR remains unchanged. (See Section 1 for descriptions of LISPEDIT commands and Section 4 for a description of LISP library files.

If a syntax error is detected during the conversion from token string to S-expressions (i.e., if the string contains unmatched parentheses or dots out of context), no filing will occur, a diagnostic will be printed, LB will be set to the point at which the error was detected, RB will be set to the tail of the string, and the remainder of the current input line will be ignored.

Table 2. STRINGED Commands

ABBREV.	NAME	ARGUMENTS
A	ADVANCE	n
B	BOUND	s
D	DELETE	-
E	ECHO	-
F	FIND	s
H	STASH	s
HK	STASHKEEP	s
I	INSERT	s
IK	INSERTKEEP	s
L	LABEL	c
N	NEXT	-
O	ONEXPR	-
P	PRINT	-
R	REPLACE	s
S	SUBSTITUTE	n s
T	TOP	-
U	OUTEXPR	-
W	WHOLE	-
X	EXTEND	n
Z	POSITION	-
-	EXIT	-
-	FILE	f

ECHO E -

The "echo switch" (initially off) is flipped, from off to on or vice versa, depending upon its current state. While the echo switch is on, a PRINT command is implied after every executed command which does not call for the printing of FR. This feature allows the editing process to be monitored step-by-step.

PRINT P -

FR is printed in full or elliptically, depending upon its length (number of tokens within it). If the label is less than 17, FR is printed in its entirety (PRINT in this case is identical to the command WHOLE). If, however, FR includes 17 or more tokens, only the first eight and last eight of these, separated by the ellipsis symbol "...", are printed. Thus, whatever the length of FR, PRINT will, in general, produce only one line to represent it. FR remains unchanged.

WHOLE W -

FR is printed in full, whatever its length may be. Should FR be empty, a blank line will appear. FR is unchanged.

POSITION Z -

A message is printed which indicates the location of LB relative to the string head, the length of FR, and the total length of the token string. The specific format of this message is:

```
          POSITION a LENGTH b TOTAL c
```

where a = number of tokens between string head and LB
b = length of FR (number of tokens therein)
c = length of the token string

FR remains unchanged.

LABEL L c

A copy of FR is saved under label c for future reference as a saved fragment argument. If a saved fragment labeled c already exists, it will be replaced by the current FR. FR remains unchanged. (See Section 2.4 for a full description of saved fragments.)

TOP T -

LB and RB are set to the string head and tail, respectively. Thus, after execution of this command, FR will include the entire token string.

NEXT N -

LB is set to the previous RB; RB is reset to the tail of the string. Thus, after execution of a NEXT command, FR becomes that part of the token string which was forward of (to the right of) the old FR. NEXT in combination with the ONEXPR command is particularly useful for skipping over S-expressions.

ONEXPR O -

RB is moved, either forward or backward, such that there will be exactly one S-expression between LB and RB; LB remains unchanged. After execution of this command, therefore, FR will contain a single S-expression.

If it is syntactically impossible to construct an S-expression beginning at LB and reading forward (i.e., if LB is at the tail of the string, if the first token forward of LB is a right parenthesis, if an out-of-context dot is encountered, etc.), a diagnostic message will be printed, LB will be set to the point at which the syntax error was detected, RN will be set to the tail of the string, and the new FR will be elliptically printed. The user is cautioned to be prepared for such a contingency.

OUTEXPR U -

OUTEXPR is identical to ONEXPR, except for the fact that, if a legitimate S-expression is found, it will be printed in its entirety (using the LISP function PRINT).

ADVANCE A n

LB is moved forward (backward) n (-n) tokens, where n is a positive (negative) integer, except that if such movement would take LB past the tail (head) of the string, then LB is set at the tail (head). RB remains unchanged unless LB is moved past (i.e., to the right of) it, in which case it is set to the tail of the string. In short, LB is moved n tokens (n positive for forward direction) with appropriate resolutions being made in impossible situations.

EXTEND X n

RB is moved forward (backward) n (-n) tokens, where n is a positive (negative) integer, except that if such movement would carry RB past the string tail (LB), then RB is set at the tail (LB); LB remains unmoved. Thus, EXTEND does to RB what ADVANCE does for LB, except that, whereas LB can be advanced past RB, RB can not be extended back through LB. In short, RB is moved n tokens (n positive for forward direction), but not past LB or the tail of the string.

DELETE D -

FR is deleted from the string, LB remains at the point of the deletion, and RB is reset to the same point. Thus, after execution of a DELETE command, FR is empty.

STASH H s

String argument s is inserted into the string at LB; FR is set to the inserted fragment. Thus, the new FR is a fragment input to the left of the old FR.

STASHKEEP HK s

This command is identical to STASH except that, after its execution, FR will contain the old FR as well as the newly inserted fragment.

INSERT I s

String argument s is inserted into the string at RB; FR is set to the inserted fragment. Thus, the new FR is a fragment input to the right of (forward of) the old FR. INSERT is equivalent to a NEXT followed by a STASH.

INSERTKEEP IK s

This command is identical to INSERT except that, after its execution, FR will contain the old FR as well as the newly inserted fragment.

REPLACE R s

The old FR is replaced by string argument s, which becomes the new FR. Thus, the new FR is a fragment input to replace the old FR. REPLACE is equivalent to a DELETE followed by INSERT or STASH.

FIND F s

Beginning at LB. a search is made in the forward direction (to the right) until a portion of the string is encountered which is identical to t he string argument s. FR is then set to that portion. Two numerical tokens are deemed identical if and only if they are equal both in value and in type (LISP predicate *EQN is used here). FIND is probably the most useful of the stringed commands for purposes of editing.

If a portion of the string identical to s is not found forward of LB, a diagnostic message will be printed, and FR will remain unchanged. The user is cautioned to be prepared for this contingency.

BOUND B s

BOUND is identical to FIND except that, where the search has been successful, instead of setting FR to merely the found fragment, FR is set to that portion of the string which begins at the original LB and extends to the right boundary of the found fragment. This may be stated even more simply--BOUND is identical to FIND except that under no circumstances is LB allowed to be moved. A particular use of BOUND in conjunction with FIND is in setting FR to a long portion of the string. FIND is first employed to locate the left boundary thereof, then a BOUND command is given to set the right boundary without having to change the left one.

SUBSTITUTE S n s

String argument S replaces FR and the next n-1 string portions identical to FR, searching to the right with FIND. Algorithmically, execution of the command involves the following loop: REPLACE current FR by a copy of s, use NEXT to move past the replacement, FIND portion identical to the original FR, repeat if FIND command was successful and decremented n is non-zero, otherwise terminate with FR set to the last replacement made. SUBSTITUTE is used generally for three purposes: multiple substitution, multiple deletion and multiple replication. For multiple substitution, the first instance of the fragment to be replaced must be located (using FIND), and then a SUBSTITUTE command given with the replacement fragment and number of replacements to be made as arguments. Multiple deletion is accomplished similarly with an empty fragment being entered as the string argument of SUBSTITUTE. For multiple replication, the point at which the copies are to be placed is located, a FIND command with an empty argument is issued (FR becomes an empty fragment situated at the prior LB of FR), and finally a SUBSTITUTE command is given with n as the number of copies of the string argument to be produced (upon termination, FR will be set to the rightmost of the copies).

If argument n is not an integer greater than zero, a diagnostic message is printed and FR remains unchanged. If a search fails at any time before n reaches zero, the action of SUBSTITUTE is effectively terminated by the action of the FIND.

3. LISPED FUNCTIONS

LISPED uses the following functions whose names are visible to the user:

```
LISPEDIT
TEDFILER
TEDSEEKER
STRINGED
```

All other functions, macros, and special variables have had their names removed both to save space and to avoid conflicts with the user.

Property lists of variables TFL, SVL and SSP are used by LISPED under the indicators SID and TED.

4. LIBRARY FILES AND DATA STRUCTURES

The data files maintained by LISPEDIT consist of a list of S-expressions (library file) stored on the property list of the atom TFL under the property TED. Each library file consists of a single S-expression whose CAR is the file name and whose CDR is a list of the contents of the file. To be compatible with RUN, RUNSPEAK and LOADEXP, the contents of a file should be a series of Evalquote pairs.

The corresponding tape format consists of one S-expression containing each library file. For example, suppose that there are two files LIB1 and LIB2 containing

```
CSET(AA 2) EVAL1 (AA)
```

and

```
DEFINE (((NILF (LAMBDA () ( ))) (ONEP (LAMBDA (J) (EQUAL J 1)))))
```

respectively. The LISP library files would appear on tape or disc as

```
(LIB1 CSET (AA 2)  EVAL1 (AA))
(LIB2 DEFINE (((NILF (LAMBDA () ( ))) (ONEP (LAMBDA (J) (EQUAL J 1))))))
End-of-file
```

After the LISPED READ command is performed, the property list of TFL contains the list

```
((LIB1 CSET (AA 2) EVAL1 (AA))
 (LIB2 DEFINE (((NILF (LAMBDA NIL NIL))
 (ONEP (LAMBDA (J) (EQUAL J 1)))))) EOF)
```

under the property TED and the value of TFL is the same list.

After the series of commands

```
INPUT
I SPECIAL ((A B C)) /
FILE LIB3 EXIT
```

the new file, (LIB3 SPECIAL ((A B C)), is spliced into the property list ahead of the EOF.

Two functions exist within LISPEDIT to provide access to the contents of files from within Evalquote mode.

TEDSEEKER (f) yields the library file ℓf as the entire S-expression whose CAR is name of the file, f.

TEDFILER (ℓf) enters the library file ℓf into the list of files under the name f given as the CAR of ℓf.

NAME	ROOM
E. WALLER	FALLS CHURCH
G. WEINWURM	2312
G. WEISBORD	4524
C. WEISSMAN	2214
G. WILEY	2059
S. WILKS	FALLS CHURCH
R. WYLLYS	9636
J. YOTT	FALLS CHURCH
M. DRAPER	2047
E. LIPNICK	4035
T. RUGGLES	9724
J. SCROGGINS	FALLS CHURCH
W. WILLIAMS	2313

DISTRIBUTION LIST

NAME	ROOM
STEPHANIE ACKLEY	2056
M. ALMQUIST	9623
S. ARANDA	2033
B. BARANCIK	2105
R. BARE	12058
J. BARNETT	2059
P. BARTRAM	2336
D. BEELER	2230
J. BURGER	9919
MYRNA BERNICK	9018
B. BICHEL	DAYTON
D. BLANKENSHIP	9517
M. BLAUER	PARAMUS
R. BLEIER	2317
M. BLEIER	2324
E. BOOK	2332
D. BORETA	2062
BOB BOSAK	2041
S. BOWMAN	2322
H. BRATMAN	1137
J. BREEN	FALLS CHURCH
R. BREWER	OMAHA
W. R. BRUSO	7120
S. BROWN	2045
A. BUMSTED	FALLS CHURCH
J. BURGER	9919
H. BURNAUGH	9630
M. CALLAHAN	4569
G. CANTLEY	FALLS CHURCH
H. CARTER	DAYTON
P. CHANEY	9917
A. CHAPMAN	8235
E. CLARK	2338
H. CLARK	2231
V. COHEN	2326
N. COLES	12058
B. CONLEY	9521
R. COOK	5178
W. COZIER	2224
B. CROSSLEY	3757
W. CUMMINS	10080
W. DENNIS	2055
P. DESIMONE	2316
B. DILLER	4025
R. DINSMORE	2220
G. DOBBS	2111
W. DOBRUSKY	2117
C. P. DONAHUE	9529
D. DRUKEY	2105

NAME	ROOM
K. MCCONLOGUE	9439
ROSALIND MCCRAKEN	8629
J. MCDONALD	23029
P. MCISAAC	2320
B. MOORE	FALLS CHURCH
W. MOORE	9111
C. MOSMANN	3730
E. MYER (51)	2227
E. NEWLANDS	4772
P. NEWMAN	COLORADO SPRINGS
M. O'CONNOR	12058
L. PAGE	5250
F. PALMER	FALLS CHURCH
S. PERLMAN	17013
D. PERRY	2060
M. PERSTEIN	2334
J. PETERKA	2060
R. PETERSON	COLORADO SPRINGS (NORAD)
F. POAGE	PARAMUS
J. B. PORGES	2063
B. REYNOLDS	2021
J. REYNOLDS	2226
D. RICHMOND	LEXINGTON
J. ROSENBAUM	3725
T. ROWAN	2175
N. SANDIN	1419
D. SAVITT	5209
M. SCHAEFER	2424
R. SCHAUB	COLORADO SPRINGS
J. SCHEID	4511
W. SCHOENE	20068
V. SCHORRE	9024
J. I. SCHWARTZ	2123
J. SCROGGINS	FALLS CHURCH
S. SHAFFER	9721
S. SHAPIRO	4364
MARY ANN SHAW	2056
H. SILBERMAN	9518
B. SIMMONS	9908
A. SKRUKRUD	COLORADO SPRINGS
J. SLAYBAUGH	8629
J. SMITH	DAYTON
G. S. STANTON (10)	PARAMUS
R. H. STEARNS	FALLS CHURCH
T. STEEL	9532
E. STEFFERUD	9734
F. STEO	2039
R. H. STERNECK	2045
K. THETING	PARAMUS
R. TOTSCHEK	9017
A. TSCHEKALOFF	2211
V. R. UNRUH	2411
A. VORHAUS	2213

NAME	ROOM
L. DURHAM	2424
PAT EDDY	2425
D. ESTAVAN	9527
J. FARELL	3753
S. FEINGOLD	9720
C. FIALA (3)	8627
LEAH FINE	2356
DONNA FIRTH	2310
B. FITZGERALD	4451
E. FOOTE	2048
C. FOX	2222
B. FREEMAN	1181
CHUCK FRYE	9523
L. GALLENSON	9920
L. GILLESPIE	5220
M. GOETSCH	7110
G. GRANT	2046
JOHN GULLAHORN	9725
D. HAGGERTY	9011
J. HALE	2048
J. HANNA	2110
H. HARMAN	9021
L. HAWKINSON (4)	9717
J. HODGSON	2314
K. A. HOLCOMB	OMAHA
S. HOLLEN	FALLS CHURCH
R. HOUSTON	4367
M. HOWARD	2060
R. HOWARD	2054
A. IRVINE	1139
H. HOWELL	9912
H. ISBITZ	22130
J. JAFFE	24143
D. JAMIESON	LEXINGTON
BART JONES	2231
S. KAMENY (50 copies)	2009
C. KELLOGG	2228
D. KEMPER	9932
PHYLLIS KENNEDY	2419
T. KOESKE	20115
T. KREBS	2054
C. KRIBS	2023
P. KRIBS	1232
B. KROUSS	9307
N. LARKS	4367
C. LAWSON	1218
R. LINDE	2229
D. LONDE	9730
R. LONG	9926
W. LUCAS	OMAHA
H. MANELOWITZ	9716
D. MARSH	9506
L.B. MCCABE	3750
J. MCCAFFERTY	LEXINGTON