File created: 19-Apr-2023 18:58:13 **{DSK}<home>larry>il>medley>sources>HIST.;6**

edit by: lmm

changes to: (FNS GREET0)

previous date: 19-Mar-2023 10:09:08 {DSK}<home>larry>il>medley>sources>HIST.;1

Read Table: XCL

Package: INTERLISP

Format: XCCS

```
(RPAQQ HISTCOMS
       ((FNS PRINTHISTORY ENTRY# PRINTHISTORY1 PRINTHISTORY2)
        (FNS EVALQT ENTEREVALQT USEREXEC LISPXREAD LISPXREADBUF LISPXREADP LISPXUNREAD LISPX LISPX/ LISPX/1
             LISPXEVAL LISPXSTOREVALUE HISTORYSAVE LISPXFIND LISPXGETINPUT REMEMBER GETEXPRESSIONFROMEVENTSPEC
             LISPXFIND0 LISPXFIND1 HISTORYFIND HISTORYFIND1 HISTORYMATCH VALUEOF VALUOF VALUOF-EVENT LISPXUSE
             LISPXUSE0 LISPXUSE1 LISPXSUBST LISPXUSEC LISPXFIX CHANGESLICE LISPXSTATE LISPXTYPEAHEAD)
        (ALISTS (SYSTEMINITVARS LISPXHISTORY GREETHIST))
        (DECLARE\: DONTEVAL@LOAD DOCOPY (VARS (\#REDOCNT 3)
                                              (ARCHIVEFLG T)
                                              (ARCHIVEFN)
                                              (ARCHIVELST '(NIL 0 50 100))
                                              (DISPLAYTERMFLG)
                                              (EDITHISTORY '(NIL 0 30 100))
                                              (HERALDSTRING)
                                              (LASTEXEC)
                                              (LASTHISTORY)
                                              (LISPXBUFS)
                                              (LISPXHIST)
                                              (LISPXHISTORY '(NIL 0 30 100))
                                              (LISPXPRINTFLG T)
                                              (LISPXUSERFN)
                                              (MAKESYSDATE)
                                              (PROMPT#FLG T)
                                              (REDOCNT)
                                              (SYSOUT.EXT 'SYSOUT)
                                              (SYSOUTFILE 'WORK)
                                              (SYSOUTGAG)
                                              (TOPLISPXBUFS)))
        (LISPXMACROS SHH RETRIEVE BEFORE AFTER OK REMEMBER\: REMEMBER TYPE-AHEAD ??T)
        (ADDVARS (LISPXFINDSPLST FROM TO THRU SUCHTHAT ALL AND)
               (BEFORESYSOUTFORMS (SETQ SYSOUTDATE (DATE))
                      (PROGN (COND ((NULL FILE)
                                    (SETQ FILE SYSOUTFILE))
                                   (T (SETQ SYSOUTFILE (PACKFILENAME 'VERSION NIL 'BODY FILE))))
                             (COND ((AND (NULL (FILENAMEFIELD FILE 'EXTENSION))
                                         (NULL (FILENAMEFIELD FILE 'VERSION)))
                                    (SETQ FILE (PACKFILENAME 'BODY FILE 'EXTENSION SYSOUT.EXT))))))
               (RESETFORMS (SETQ READBUF NIL)
                      (SETQ READBUFSOURCE NIL)
                      (SETQ TOPLISPXBUFS (OR (CLBUFS T)
                                             TOPLISPXBUFS))
                      (COND ((EQ CLEARSTKLST T)
                             (COND ((EQ NOCLEARSTKLST NIL)
                                    (CLEARSTK))
                                   (T (* |clear| |all| |stack| |pointers| EXCEPT |those| |on| NOCLEARSTKLST.)
                                      (MAPC (CLEARSTK T)
                                            (FUNCTION (LAMBDA (X)
                                                              (AND (NOT (FMEMB X NOCLEARSTKLST))
                                                                   (RELSTK X))))))))
                            (T (MAPC CLEARSTKLST (FUNCTION RELSTK))
                               (SETQ CLEARSTKLST NIL))))
               (HISTORYSAVEFORMS)
               (LISPXCOMS Ÿ |...| ?? FIX FORGET NAME ORIGINAL REDO REPEAT RETRY UNDO USE |fix| |forget| |name|
                      |redo| |repeat| |retry| |undo| |use|)
               (SYSTATS (LISPXSTATS LISPX INPUTS)
                      (UNDOSAVES UNDO SAVES)
                      (UNDOSTATS CHANGES UNDONE)
                      NIL
                      (EDITCALLS CALLS TO EDITOR)
                      (EDITSTATS EDIT COMMANDS)
                      (EDITEVALSTATS COMMANDS INVOLVING EVALUATING A LISP EXPRESSION)
                      (EDITESTATS USES OF AN E COMMAND TYPED IN DIRECTLY)
                      (EDITISTATS USES OF AN I COMMAND TYPED IN DIRECTLY)
                      (EDITUNDOSAVES EDIT UNDO SAVES)
                      (EDITUNDOSTATS EDIT CHANGES UNDONE)
                      NIL
                      (P.A.STATS P.A. COMMANDS)
                      NIL
                      (CLISPIFYSTATS CALLS TO CLISPIFY)
                      NIL
                      (FIXCALLS CALLS TO DWIM)
                      (FIXTIME)
                      (ERRORCALLS WERE DUE TO ERRORS)
                      (DWIMIFYFIXES WERE FROM DWIMIFYING)
```

```
                    NIL "OF THOSE DUE TO ERRORS:" (TYPEINFIXES WERE DUE TO ERRORS IN TYPE-IN)
                    (PROGFIXES WERE DUE TO ERRORS IN USER PROGRAMS)
                    (SUCCFIXES1 OF THESE CALLS WERE SUCCESSFUL)
                    NIL "OF THE CALLS DUE TO DWIMIFYING:" (SUCCFIXES2 WERE SUCCESSFUL)
                    NIL
                    (SPELLSTATS OF ALL DWIM CORRECTIONS WERE SPELLING CORRECTIONS)
                    (CLISPSTATS WERE CLISP TRANSFORMATIONS)
                    (INFIXSTATS OF THESE WERE INFIX TRANSFORMATIONS)
                    (IFSTATS WERE IF/THEN/ELSE STATEMENTS)
                    (I.S.STATS WERE ITERATIVE STATEMENTS)
                    (MATCHSTATS WERE PATTERN MATCHES)
                    (RECORDSTATS WERE RECORD OPERATIONS)
                    NIL
                    (SPELLSTATS1 OTHER SPELLING CORRECTIONS\, E.G. EDIT COMMANDS)
                    NIL
                    (RUNONSTATS OF ALL SPELLING CORRECTIONS WERE RUN-ON CORRECTIONS)
                    NIL
                    (VETOSTATS CORRECTIONS WERE VETOED)
                    NIL)
            (NOCLEARSTKLST))
    (APPENDVARS (AFTERSYSOUTFORMS (COND ((LISTP SYSOUTGAG)
                                         (EVAL SYSOUTGAG))
                                        (SYSOUTGAG)
                                        ((OR (NULL USERNAME)
                                             (EQ USERNAME (USERNAME NIL T)))
                                         (TERPRI T)
                                         (PRIN1 HERALDSTRING T)
                                         (TERPRI T)
                                         (TERPRI T)
                                         (GREET0)
                                         (TERPRI T))
                                        (T (LISPXPRIN1 '"****ATTENTION USER " T)
                                           (LISPXPRIN1 (USERNAME)
                                                  T)
                                           (LISPXPRIN1 '":
                                                       this sysout is initialized for user " T)
                                           (LISPXPRIN1 USERNAME T)
                                           (LISPXPRIN1 '".
                                                       " T)
                                           (LISPXPRIN1 '"To reinitialize, type GREET()
                                                       " T)))
                        (SETINITIALS)))
    (P (MAPC SYSTATS (FUNCTION (LAMBDA (X)
                                   (AND (LISTP X)
                                        (EQ (GETTOPVAL (CAR X))
                                            'NOBIND)
                                        (SETTOPVAL (CAR X)
                                               NIL)))))
       (PUTD 'E))
    (COMS (FNS GREET GREET0)
          (ADDVARS (PREGREETFORMS (DREMOVE GREETFORM RESETFORMS)
                                  (SETQ CONSOLETIME (SETQ CPUTIME (SETQ EDITIME 0)))
                                  (SETQ CONSOLETIME0 (CLOCK 0))
                                  (SETQ CPUTIME0 (CLOCK 2)))
                   (POSTGREETFORMS (SETINITIALS)
                                   (AND EDITCHARACTERS (APPLY 'SETTERMCHARS EDITCHARACTERS))))
          (DECLARE\: DONTEVAL@LOAD DOCOPY (VARS (GREETHIST)
                                                (SYSTEMTYPE)
                                                (GREETFORM '(LISPXEVAL '(GREET)
                                                                       '_))
                                                (CUTEFLG)
                                                (GREETDATES '((" 1-JAN" . "Happy new year")
                                                              ("12-FEB" . "Happy Lincoln's birthday")
                                                              ("14-FEB" . "Happy Valentine's day")
                                                              ("22-FEB" . "Happy Washington's birthday")
                                                              ("15-MAR" . "Beware the Ides of March")
                                                              ("17-MAR" . "Happy St. Patrick's day")
                                                              ("18-MAY" . "It's Victoria Day")
                                                              (" 1-JUL" . "It's Canada Day")
                                                              ("31-OCT" . "Trick or Treat")
                                                              (" 5-NOV" . "<boom> it's Guy Fawkes day")
                                                              ("25-DEC" . "Merry Christmas")))
                                                (USERNAME)
                                                (HOSTNAME)
                                                (CONSOLETIME 0)
                                                (CONSOLETIME0 0)
                                                (CPUTIME 0)
                                                (CPUTIME0 0)
                                                (EDITIME 0)
                                                (FIRSTNAME))
                   (ADDVARS (BEFOREMAKESYSFORMS (SETQ RESETFORMS (CONS GREETFORM RESETFORMS))
                                                (SETQ MAKESYSDATE (DATE))))
                   (ADDVARS (AFTERMAKESYSFORMS (LISPXEVAL '(GREET)
                                                          '_)))))
    (FNS LISPXPRINT LISPXPRIN1 LISPXPRIN2 LISPXPRINTDEF LISPXPRINTDEF0 LISPXSPACES LISPXTERPRI LISPXTAB
         USERLISPXPRINT LISPXPUT)
    (GLOBALVARS \#REDOCNT ARCHIVEFLG ARCHIVEFN ARCHIVELST BOUNDPDUMMY BREAKRESETVALSLST CAR/CDRNIL CHCONLST1
```

```
              CLEARSTKLST CLISPARRAY CLISPCHARS CLISPFLG CLISPTRANFLG CONSOLETIME CONSOLETIME0 CPUTIME CPUTIME0
              CTRLUFLG CUTEFLG DISPLAYTERMFLG DWIMFLG EDITHISTORY EDITIME EDITQUIETFLG EDITSTATS EVALQTFORMS
              FILERDTBL FIRSTNAME GREETDATES GREETHIST HISTORYCOMS HISTORYSAVEFN HISTORYSAVEFORMS HISTSTR0
              HISTSTR2 HISTSTR3 IT LASTHISTORY LISP-RELEASE-VERSION LISPXBUFS LISPXCOMS LISPXFINDSPLST LISPXFNS
              LISPXHISTORY LISPXHISTORYMACROS LISPXMACROS LISPXPRINTFLG LISPXREADFN LISPXSTATS LISPXUSERFN
              MACSCRATCHSTRING NEWUSERFLG P.A.STATS POSTGREETFORMS PREGREETFORMS PRETTYHEADER RANDSTATE
              READBUFSOURCE REDOCNT REREADFLG RESETFORMS SYSFILES TOPLISPXBUFS USERHANDLE USERNAME)
       (VARS (LISP-RELEASE-VERSION 2.0))
       (BLOCKS (LISPXFINDBLOCK LISPXFIND LISPXFIND0 LISPXFIND1 HISTORYFIND HISTORYFIND1 (ENTRIES LISPXFIND
                                                                                          HISTORYFIND)
                  (LOCALFREEVARS _FLG L LST Z =FLG HISTORYFLG PREDFLG LINE HISTORY TYPE BACKUP QUIETFLG)
                  (NOLINKFNS HISTORYMATCH LISPXGETINPUT))
              (NIL ENTRY# EVALQT GETEXPRESSIONFROMEVENTSPEC GREET GREET0 HISTORYMATCH HISTORYSAVE LISPX LISPX/
                   LISPX/1 LISPXEVAL LISPXFIND1 LISPXGETINPUT LISPXPRIN1 LISPXPRIN2 LISPXPRINT LISPXPRINTDEF
                   LISPXPRINTDEF0 LISPXPUT LISPXREAD LISPXREADBUF LISPXREADP LISPXSPACES LISPXSTOREVALUE
                   LISPXSUBST LISPXTAB LISPXTERPRI LISPXTYPEAHEAD LISPXUNREAD LISPXUSE LISPXUSE0 LISPXUSE1
                   LISPXUSEC PRINTHISTORY PRINTHISTORY1 PRINTHISTORY2 USEREXEC USERLISPXPRINT VALUEOF VALUOF
                   (LOCALVARS . T)
                   (SPECVARS LISPXLINE LISPXID LISPXVALUE LISPXLISTFLG HISTORY ID EVENT BREAKRESETVALS VARS
                          GENLST INITLST NAME MESSAGE)
                   (LINKFNS . T)
                   (NOLINKFNS LISPXTYPEAHEAD UNDOLISPX ARCHIVEFN LISPXFIX LISPXUSE LISPXUSE0 LISPXSUBST
                          LISPXFIND HISTORYMATCH PRINTHISTORY DISPLAYTERMP LISPXSTOREVALUE HISTORYSAVEFN
                          ENTEREVALQT PRINTHISTORY1 PRINTHISTORY2 LISPXFIND HISTORYMATCH LISPXGETINPUT
                          LISPXSUBST ARCHIVEFN LISPXFIX LISPXUSE LISPXUSE0 LISPXSUBST HISTORYMATCH PRINTHISTORY
                          DISPLAYTERMP LISPXSTOREVALUE HISTORYSAVEFN ENTEREVALQT LISPXTYEAHEAD UNDOLISPX
                          GREETFILENAME)))
       (DECLARE\: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA VALUEOF)
                                                                              (NLAML)
                                                                              (LAMA)))))

(DEFINEQ
```

(**PRINTHISTORY**

```
  (LAMBDA (HISTORY LINE SKIPFN NOVALUES FILE)                    (* |wt:| 7-MAY-76 4 58)
    (AND (EQ HISTORY EDITHISTORY)
         (SETQ NOVALUES T))

          (* NOVALUES |is| T |for| |printing| EDITHISTORY\, |indicates| |not| |to| |print| |the| |value.|
          |if| |it| |is| |non-atomic,| |it| |is| \a |form| |which| |is| |evaluated| |in| PRINTHISTORY1 |in| |lieu| |of| |printing| |the| |value.|
          |This| |form| |can| |also| |be| |obtained| |from| |the| |property| |list| |of| |the| |entry| |under| |property| |print.|)

    (PROG ((L (CAR HISTORY))
           LST HELPCLOCK)
          (SETQ LST (COND
                       ((NULL LINE)
                        (CAR HISTORY))
                       (T (LISPXFIND HISTORY LINE 'ENTRIES))))
          (TERPRI FILE)
          (TERPRI FILE)
          (MAPC LST (FUNCTION (LAMBDA (EVENT)
                                (COND
                                   ((AND SKIPFN (APPLY* SKIPFN EVENT))
                                                                  (* |\f SKIPFN |applied| |to| |this| |entry| |is| T\, |it| |is| |skipped.|)
                                    )
                                   (T (PRIN2 (ENTRY# HISTORY EVENT)
                                             FILE T)
                                      (PRIN1 '\. FILE)
                                      (PRINTHISTORY1 EVENT (COND
                                                              ((EQ NOVALUES T)
                                                               T)
                                                              (NOVALUES (APPLY* NOVALUES EVENT)))
                                             FILE))))))
          (TERPRI FILE)
          (TERPRI FILE)
          (RETURN))))
```

(**ENTRY#**

```
  (LAMBDA (HIST X)
    (COND
       ((NOT (IGREATERP (SETQ X (IPLUS (CADR HIST)
                                       (IMINUS (FLENGTH (CAR HIST)))
                                       (FLENGTH (FMEMB X (CAR HIST)))))
                        0))
        (IPLUS X (OR (CADDDR HIST)
                     100)))
       (T X))))
```

(**PRINTHISTORY1**

```
  (LAMBDA (EVENT NOVALUES FILE)                                  (* |lmm| " 1-May-86 13:56")
                                                                 (* |\f NOVALUES |is| T\, |means| |suppress| |printing| |of| |value.|)
    (PROG ((INPUT (CAR EVENT))
           Y TEM)
          (COND
             ((LISTP (SETQ TEM (LISTGET1 EVENT '*FIRSTPRINT*)))  (* |used| |by| |the| |editor.|)
```

```
            (TAB 5 NIL FILE)
            (APPLY (CAR TEM)
                   (CONS FILE (CDR TEM)))))
         (COND
            ((SETQ Y (CDR (FMEMB '*GROUP* EVENT)))

         (* MEMB |used| |instead| |of| LISTGET |because| |value| |may| |be| NIL\, |e.g.|
         |if| |command| |aborted| |because| USE |argument| |wasnt| |found.|)

            (TAB 5 NIL FILE)
            (MAPRINT (LISTGET1 EVENT '*HISTORY*)
                   FILE NIL NIL NIL (FUNCTION (LAMBDA (X FL)
                                              (PRIN2 X FL T))))
            (TERPRI FILE)
            (COND
               ((CAR Y)
                (MAPC (CAR Y)
                      (FUNCTION (LAMBDA (EVENT)
                                  (PRINTHISTORY1 EVENT NOVALUES FILE))))
                (COND
                   ((SETQ TEM (LISTGET1 EVENT '*REDOCNT*))
                    (TAB 5 NIL FILE)
                    (PRIN1 "... " FILE)
                    (PRIN1 (ADD1 TEM)
                           FILE)
                    (PRIN1 " times
                           " FILE)))
                (RETURN)

         (* |if| |group| |is| |empty,| |still| |might| |want| |to| |drop| |through| |and| |print| |input,| |if| |any,| |e.g.|
         NAME |command| |works| |this| |way.|)

               ))))
         (COND
            ((OR (NULL INPUT)
                 (EQ (CAR INPUT)
                     HISTSTR2))
             (GO LP1)))
         (TAB 5 NIL FILE)
         (AND (SETQ TEM (CADR EVENT))
              (PRIN1 TEM FILE))
   LP    (COND
            ((SETQ Y (FMEMB HISTSTR0 (LISTP INPUT)))
             (SETQ INPUT (LDIFF INPUT Y))))
         (AND INPUT (PRINTHISTORY2 INPUT FILE NOVALUES))

         (* |shouldnt| |be| |any| |situations| |with| |two| "<c.r.>" \s |in| \a |row,| |but| |just| |in| |case|)

         (COND
            (Y (SETQ INPUT (CDR Y))
               (SPACES 5 FILE)
               (GO LP)))
   LP1   (MAPC (LISTGET1 EVENT '*LISPXPRINT*)
               (FUNCTION (LAMBDA (X)
                           (LISPXREPRINT X FILE))))
         (COND
            ((LISTP (SETQ TEM (LISTGET1 EVENT '*PRINT*)))       (* |used| |by| |break.|)
             (TAB 5 NIL FILE)
             (APPLY (CAR TEM)
                    (CONS FILE (CDR TEM))))
            (NOVALUES)
            (T (|for| X |in| (LISTGET (CDDDR EVENT)
                                      'LISPXVALUES)
                  |do| (TAB 5 NIL FILE)
                       (SHOWPRINT X FILE T)))))))
```

(PRINTHISTORY2

```
  (LAMBDA (INPUT FILE NOVALUES)                                  (* |wt:| "14-AUG-78 02:59")
    (PROG (TEM)
          (COND
             ((NLISTP INPUT)
              (PRIN1 INPUT FILE))
             ((CDDR INPUT)
              (MAPRINT INPUT FILE NIL NIL NIL (FUNCTION (LAMBDA (X FL)

          (* MAPRINT |does| |an| |apply*| |with| |this| |argument| |on| |the| |thing| |to| |be| |printed| |and| |the| |fl.|)

                                                          (SHOWPRIN2 X FL T)))))
             ((CDR INPUT)                                        (* APPLY |input|)
              (SHOWPRIN2 (CAR INPUT)
                     FILE T)
              (COND
                 ((NULL (SETQ TEM (CADR INPUT)))
                  (PRIN1 (COND
                             (\#RPARS ']')
                             (T '|()|))
```

```
                    FILE))
               (T (COND
                     ((OR (ATOM TEM)
                          (EQ NOVALUES T))                    (* I\f NOVALUES |is| T\, |ppobaby| |is| |printing| |editor| |history|
                                                                 |list,| |so| |print| |the| |space.|)
                      (SPACES 1 FILE)))
                  (SHOWPRIN2 TEM FILE T))))
            (T                                                (* EVAL |input|)
               (SHOWPRIN2 (CAR INPUT)
                     FILE T)))
         (TERPRI FILE))))

)

(DEFINEQ
```

(**EVALQT**

```
  (LAMBDA (LISPXID)                                           (* |lmm| " 9-Jun-85 21:04")
    (PROG NIL
          (COND
             ((NULL LISPXID)
              (SETQQ LISPXID _)
              (ENTEREVALQT)))
          (FRESHLINE T)
      LP  (PROMPTCHAR LISPXID T LISPXHISTORY)
          (COND
             ((NULL (ERSETQ (LISPX (LISPXREAD T T)
                                   LISPXID)))
              (SETQ TOPLISPXBUFS (OR (CLBUFS T)
                                     TOPLISPXBUFS))
              (TERPRI T)))

          (* |this| |errorset| |is| |so| |that| EVALQTFORMS |dont| |get| |unnecessarily| |evaluated| |following| |each| |error| |on| |typein.|
          |they| |are| |only| |for| |control-d.|)

          (GO LP))))
```

(**ENTEREVALQT**

```
  (LAMBDA NIL                                                 (* |lmm| " 7-Nov-86 03:47")
    ;; this is not on resetforms, because it is important that it be done first, i.e.  before the form specified on resetforms.
    ;;  with unwinders it is mainly unnecessary
    ;; with multiple execs, it is probably wrong
    (RESETRESTORE NIL 'RESET)
    (MAPC RESETFORMS (FUNCTION (LAMBDA (X)
                                 (ERSETQ (EVAL X)))))))
```

(**USEREXEC**

```
  (LAMBDA (LISPXID LISPXXMACROS LISPXXUSERFN)                 (* |Pavel| " 7-Jul-86 11:26")
                                                              (* |wt:| 28-JUL-77 22 1)
    (RESETVARS (READBUF READBUFSOURCE REREADFLG)
               (CL:WHEN (NULL LISPXID)
                   (SETQ LISPXID '_))
         LP    (CL:WHEN (> (POSITION T)
                           0)
                      (CL:TERPRI T))
               (PROMPTCHAR LISPXID T LISPXHISTORY)
               (ERSETQ (LISPX (LISPXREAD T T)
                              LISPXID LISPXXMACROS LISPXXUSERFN))
               (GO LP))))
```

(**LISPXREAD**

```
  (LAMBDA (FILE RDTBL)                                        (* AJB "16-Jul-85 15:54")

          (* * A |generalized| READ. I\f READBUF |is| NIL\, |performs| |and| APPLY* LISPXREADFN FILE.
          |which| |it| |returns| |as| |its| |value.| I\f READBUF |is| |not| NIL\, "reads" |and| |returns| |the| |next| |expression| |on|
          READBUF)

    (PROG (X)
      LP  (COND
             ((NULL (AND READBUF (SETQ READBUF (LISPXREADBUF READBUF T))))
              (SETQ REREADFLG NIL)
              (SETQ X (COND
                         ((OR (EQ LISPXREADFN 'READ)
                              (IMAGESTREAMTYPEP T 'TEXT))     (* S\o |the| |call| |will| |be| |linked,| |so| |the| |user| |can| |break|
                                                                 |on| |read.|)
                          (READ FILE RDTBL))
                         (T (APPLY* LISPXREADFN FILE RDTBL))))
              (COND
                 ((AND (LISTP X)
                       CTRLUFLG)                              (* |User| |typed| |control-u| |during| |read.|
                                                                 |The| |assemble| |is| |an| OPENR.)
                  (SETQ CTRLUFLG NIL)
```

```
                (COND
                   ((NULL (NLSETQ (EDITE X)))                    (* |Exited| |with| STOP\, |just| |save| |input| |but| |do| |not|
                                                                 |evaluate| |or| |execute.|)
                    (SETQ REREADFLG 'ABORT)))))
             (RETURN X)))

          (* REREADFLG |is| |later| |used| |to| |compare| |with| |the| |first| |entry| |on| |the| |history| |list| |to| |see| |if| |the| |reread|
          |expression| CAME |from| |that| |entry.|)

          (SETQ X (CAR READBUF))
          (SETQ READBUF (CDR (SETQ REREADFLG READBUF)))
          (RETURN X))))
```

(LISPXREADBUF

```
(LAMBDA (RDBUF STRIPSEPRSFLG)                                    (* |lmm| " 4-NOV-82 23:59")

        (* |takes| |care| |of| 'cleaning'| |up| |read| |buffer| |by| |stripping| |off| |extra| "<c.r.>" |and| |processing| |repeated| |reads.|
        |used| |by| |promptchar,| |editor,| |lispxread,| |etc.|)

   (PROG (TEM)
     LP  (COND
            ((NLISTP RDBUF)
             (RETURN NIL))
            ((EQ (CAR RDBUF)
                 HISTSTR0)                                       (* HISTSTR0 |is| \a |delimiter| |for| |eadline|)
             (SETQ RDBUF (CDR RDBUF))
             (GO LP))
            ((EQ (CAR RDBUF)
                 HISTSTR3)                                       (* HISTSTR3 |is| \a |marker| |for| |flagging| |the| |event| |that|
                                                                 |the| |readbuf| |came| |from|)
             (SETQ RDBUF (CDDR RDBUF))
             (GO LP))
            ((EQ (CAR RDBUF)
                 HISTSTR2))
            (T (RETURN RDBUF)))
          (SETQ REDOCNT (ADD1 REDOCNT))
          (SETQ RDBUF (CDR RDBUF))
          (SETQ RDBUF (COND
                         ((SETQ TEM (COND
                                       ((NUMBERP (CAR RDBUF))
                                        (AND (IGREATERP (CAR RDBUF)
                                                        0)
                                             (SUB1 (CAR RDBUF))))
                                       ((EVAL (CAR RDBUF))
                                        (CAR RDBUF))))
                          (NCONC (|for| XX |in| (CADR RDBUF) |until| (EQ XX HISTSTR2)
                                    |collect| (COND
                                                 ((NLISTP XX)
                                                  XX)
                                                 (T (COPY XX))))
                                 (CONS HISTSTR0 (CONS HISTSTR2 (CONS TEM (CDR RDBUF))))))
                         ((LISPXREADBUF (CDDR RDBUF)))
                         (T (PRIN1 REDOCNT T)
                            (PRIN1 " repetitions.
                                   " T)
                            NIL)))
          (GO LP))))
```

(LISPXREADP

```
(LAMBDA (FLG)                                                    (* |lmm| " 5-NOV-82 00:00")

        (* FLG |corresponds| |to| |the| FLG |argument| |to| READP\, |i.e.|
        |if| FLG=NIL\, |returns| NI\l |if| |just| \a |c.r.| |waiting.| |if| FLG=T\, |returns| T |if| |anything| |waiting|)

   (COND
      ((AND READBUF (SETQ READBUF (LISPXREADBUF READBUF)))
       T)
      ((READP T T)
       (OR FLG (NEQ (PEEKC T)
                    (CONSTANT (CHARACTER (CHARCODE EOL)))))))))
```

(LISPXUNREAD

```
(LAMBDA (LST EVENT)                                              (* |lmm| " 5-NOV-82 00:02")
  (SETQ READBUF (APPEND LST (COND
                               (EVENT (CONS HISTSTR3 (CONS EVENT READBUF)))
                               (T (CONS HISTSTR0 READBUF))))))
```

(LISPX

```
(LAMBDA (LISPXX LISPXID LISPXXMACROS LISPXXUSERFN LISPXFLG)      (* |lmm| "11-Jul-86 18:01")

        (* LISPX (|for| LISP |eXec|) |is| |designed| |to| |save| |the| |user| |the| |task| |of| |writing| |an| |exec| |by| |allowing| |him| |to|
        |easily| |tailor| LISPX |to| |his| |applications.| I\n |this| |way,| |the| |user| |also| |gets| |the| |benefit| |of| |the| |history| |features|
        |built| |into| LISPX. LISPX |determines| |the| |type| |of| |input,| |performs| |any| |extra| |reads| |that| |are| |necessary,| |saves|
        |the| |input| (\s) |and| |the| |value| |on| |the| |history,| |and| |prints| |and| |returns| |the| |value.|
```

(LISPX |must| |do| |the| |printing| |since| |for| |history| |commands,| |see| |below,| |nothing| |can| |be| |printed| |until| |the| |next| |call| |to| LISPX.) -
-
|There| |are| |currently| |six| |different| |classes| |of| |inputs:| (1) EVAL\, |i.e.|
|forms;| (2) APPLY\, |i.e.| |functions| |and| |arguments;| (3) |forms| |without| |parentheses,| |i.e.|
|lines,| |usually| |specifying| CLISP |transformation,| |e.g.| FOR X IN |...|
I\n |this| |case| |the| |entire| |line| |is| |treated| |as| \a |form| |and| |EVALed;|
(4) |commands,| |similar| |to| |edit| |macros,| |definitions| |are| |looked| |up| |on| LISPXMACROS\;
(5) |user| |input,| |as| |determined| |by| |applying| LISPXUSERFN.
I\f |this| |yields| T\, |the| |value| |of| |the| |event| |is| |the| |value| |of| LISPXVALUE\, |which| |must| |be| |set| |by| LISPXUSERFN\; |and| (6) |history| |commands.| -
|For| |types| 1 |thru| 5\, LISPX |saves| |the| |inputs| |on| |the| |history| |list| |before| |executing.|
|Thus| |even| |if| |the| |operation| |is| |aborted,| |the| |user| |can| |redo| |it,| |fix| |it,| |etc.|
-
|For| |commands| 1\, 2\, |and| 3\, |the| |function| |name| |is| |looked| |up| |on| LISPXFNS.
|if| |the| |user| |simply| |wants| \a |different| |function| |called| |for| |tty| |inputs| |then| |in| |his| |program,| |such| |as| |is| |the| |case| |with| SETQ |or| SET\, |this| |can| |easily| |be| |done| |by| |putting| (|fn1| . |fn2|) |on| |the| |list| LISPXFNS. -
|For| |commands| |of| |type| 6\, LISPX |simply| |unreads| |the| |appropriate| |information| |and| |exits.|
|This| |means| |that| |if| \a |user| |function| |calls| LISPX |when| |it| |cannot| |interpret| |the| |input,| |history| |operations| |will| |work| |provided| |only| |that| |the| |user| |function| |obtains| |its| |input| |via| LISPXREAD\, |and| |that| |any| |inputs| |interpreted| |by| |the| |user| |function| |also| |save| |the| |input| |on| |the| |history| |list.|
|This| |is| |the| |way| BREAK1 |uses| LISPX.)

(* I\f LISPXFLG |is| T\, |any| |history| |commands| |are| |executed| |in| |this| |call| |to| LISPX\, |instead| |of| |unreading| |and| |exiting.| |This| |is| |used| |when| |the| |calling| |function| |knows| |that| |the| |input| |should| (|must|) |be| |processed| |here,| |for| |example,| |in| |the| E |command| |from| |the| |editor.|
|Without| |this,| E REDO |would| |cause| |the| |input| |referred| |to| |by| |the| REDO |command| |to| |be| |interpreted| |as| |edit| |commands| |instead| |of| LISPX |inputs.| I\f LISPXFLG |is| |'RETRY,| CLOCK |is| |backed| |up| |to| |force| \a BREAK |on| |any| |error.|)

```
(AND (NULL LISPXXMACROS)
     (SETQ LISPXXMACROS LISPXMACROS))
(AND (NULL LISPXXUSERFN)
     LISPXUSERFN
     (FGETD 'LISPXUSERFN)
     (SETQQ LISPXXUSERFN LISPXUSERFN))
```

(* I\f LISPX |is| |called| |with| |its| |fifth| |argument,| LISPXXUSERFN\, |non-NIL,| |it| |is| |applied| (|with| APPLY*)\. |Otherwise,| |the| |top| |level| |value| |of| LISPXUSERFN |is| |checked,| |and| |if| |non-NIL,| LISPXUSERFN |itself| |is| |called.| (|The| |former| |is| |for| |calls| |from| USEREXEC\, |the| |latter| |corresponds| |to| |the| |old| |way| |of| |doing| |it.| |Similarly,| |if| LISPX |is| |called| |with| |its| |fourth| |argument,| LISPXXMACROS\, |non-NIL,| |it| |is| |used| |as| |the| |list| |of| |macros,| |otherwise| |the| |top| |level| |value| |of| LISPXMACROS |is| |used.|))

```
(PROG ((HELPCLOCK (CLOCK 2))
       LISPXOP LISPXLISTFLG LISPXLINE (LISPXHIST LISPXHIST)
       LISPY LISPZ LISPXVALUE LISPXTEM DONTSAVEFLG (HELPFLAG (COND
                                                              ((EQ HELPFLAG 'BREAK!)
                                                               (* |so| |that| |when| |you| |get| |in| |the| |break,| |doesnt| |always|
                                                                  |break| |below| |that|)
                                                               (GETTOPVAL 'HELPFLAG))
                                                              (T HELPFLAG)))
       LISPXVALUES)
      (DECLARE (SPECVARS HELPFLAG LISPXVALUE LISPXVALUES))
      (COND
         ((NULL LISPXX)                                       (* |Spurious| |right| |parentheses| |or| |bracket.|)
          (RETURN (PRINT NIL T)))
         ((NLISTP LISPXX)
          (SETQ LISPXLINE (READLINE T (LIST LISPXX)
                                    T))
```

(* |The| |third| |argument| |specifies| |that| |if| |there| |is| |juut| \a "]" |or| ")" |on| |the| |line,| |it| |should| |be| |read| |as| \a NIL\, |i.e.| |the| |line| |should| |be| (NIL)\. I\t |also| |specifies| |that| |if| |the| |line| |begins| |with| \a |list| |which| |is| |not| |preceded| |by| |any| |spaces,| |the| |list| |is| |to| |terminate| |the| |line| |regardless| |of| |whether| |or| |not| |it| |is| |terminated| |by| \a].
|Thus| |the| |usr| |can| |type| |fn| (|args|))

```
          (SETQ LISPXX (CAR LISPXLINE))
          (SETQ LISPXLINE (CDR LISPXLINE))
```

(* |done| |this| |way| |so| |control-W| |will| |work| |on| |first| |thing| |read| |from| |inside| |of| |the| |readline.|)

```
         )
         ((AND (NULL REREADFLG)
               (NOT (SYNTAXP (SETQ LISPXTEM (CHCON1 (LASTC T)))
                             'RIGHTPAREN T))
               (NOT (SYNTAXP LISPXTEM 'RIGHTBRACKET T))
               (CDR (SETQ LISPXLINE (READLINE T (LIST LISPXX)
                                              T))))
```

(* |The| |expression| |input| |was| \a |lis,| |although| |it| |was| |not| |terrnated| |with| \a |right| |parent| |or| |bracket,| |e.g.| (QUOTE ZAP\,) |and| |furthermore| |there| |was| |something| |else| |on| |the| |same| |line,| |so| |treat| |it| |as| |line| |input.|
|This| |enables| |user| |to| |type| (QUOTE FOO) :EXPR)

```
          (SETQ LISPXX LISPXLINE)))
   TOP (COND
          ((LISTP LISPXX)
           (SETQ LISPXOP (CAR LISPXX))
```

```
            (SETQ LISPXLINE (CDR LISPXX))

        (* |This| |is| |for| |convenience| |of| |history| |commands:| |regardless| |of| |whether| |the| |command| |was| |typed| |as| \a |list|
        |or| \a |line,| LISPXOP |is| |always| |the| |name| |of| |the| |command,| LISPXLINE |its| |'arguments'.|
        I\f |it| |turns| |out| |that| LISPXOP |is| |not| \a |history| |command,| LISPXLINE |will| |be| |set| |back| |to| NIL
        (|below| NOTCOM))

            (SETQ LISPXLISTFLG T))
           ((NOT (LITATOM LISPXX))
            (GO NOTCOM)                                          (* |User| |might| |have| |typed| |in| \a |number| |followed| |by|
                                                                 |something| |else|)
            )
           (T (SETQ LISPXOP LISPXX)))
    SELECT
        (COND
           ((AND REREADFLG (EQ (SETQ LISPXTEM (CAR (LISTGET1 (CAAR LISPXHISTORY)
                                                              '*HISTORY*)))
                               'ORIGINAL)))
           ((SETQ LISPY (FASSOC LISPXOP LISPXXMACROS))
            (AND LISPXLISTFLG (SETQ LISPXLINE NIL))              (* |so| |historysave| |at| DO-IT |will| |get| |called| |with| |the| |right|
            |aaguments.|)
            (SETQ DONTSAVEFLG (NULL (CADR LISPY)))
            (GO DO-IT))
           ((SETQ LISPY (FASSOC LISPXOP LISPXHISTORYMACROS))
            (SETQ DONTSAVEFLG (NULL (CADR LISPY)))
            (GO REDOCOM)))
        (SELECTQ LISPXOP
            (ORIGINAL (GO REDOCOM))
            (E (COND
                  ((NULL LISPXLINE)
                   (GO NOTCOM)))
               (SETQ LISPXX (SETQ LISPXOP (CAR LISPXLINE)))
               (SETQ LISPXLINE (CDR LISPXLINE))
               (GO NOTCOM))
            ((RETRY REDO REPEAT FIX USE |...| Ÿ |redo| |repeat| |use| |fix| |retry|)
               (GO REDOCOM))
            ((|name| NAME)
               (COND
                  ((NULL LISPXLINE)                              (* T\o |allow| |user| |to| |have| NAME |as| |the| |name| |of| \a
                                                                 |variable.|)
                   (GO DO-IT)))
               (GO REDOCOM))
            ((UNDO |undo|)
               (AND (SETQ LISPXHIST (HISTORYSAVE LISPXHISTORY LISPXID NIL LISPXOP LISPXLINE))
                    (FRPLACA (CDDR LISPXHIST)
                             (UNDOLISPX LISPXLINE))))
            ((|retry:| RETRY\:)
               (AND (EQ REREADFLG 'ABORT)
                    (ERROR!))
               (SETQ HELPFLAG 'BREAK!)
               (SETQ LISPXX (CAR LISPXLINE))
               (SETQ LISPXLINE (CDR LISPXLINE))
               (GO TOP))
            ((|forget| FORGET)
               (AND (EQ REREADFLG 'ABORT)
                    (ERROR!))
               (MAPC (COND
                        (LISPXLINE (LISPXFIND LISPXHISTORY LISPXLINE 'ENTRIES))
                        (T (CAR LISPXHISTORY)))
                     (FUNCTION (LAMBDA (X)
                                  (UNDOLISPX2 X T))))
               (PRINT '|forgotten| T T))
            (?? (AND (EQ REREADFLG 'ABORT)
                     (ERROR!))
                (PRINTHISTORY (COND
                                 ((EQ (CAR LISPXLINE)
                                      '@@)
                                  (SETQ LISPXLINE (CDR LISPXLINE))
                                  ARCHIVELST)
                                 (T LISPXHISTORY))
                              LISPXLINE NIL NIL T))
            ((|archive| ARCHIVE)
               (AND (EQ REREADFLG 'ABORT)
                    (ERROR!))

        (* |Since| |these| |the| |commands| |do| |not| |call| HISTORYSAVE\, |we| |must| |check| |for| |control-U| |followed| |by| STOP
        |here.|)

               (COND
                  (ARCHIVELST (FRPLACA ARCHIVELST (NCONC (SETQ LISPXTEM (LISPXFIND LISPXHISTORY LISPXLINE
                                                                                  'COPIES))
                                                         (CAR ARCHIVELST)))
                              (FRPLACA (CDR ARCHIVELST)
                                       (IPLUS (CADR ARCHIVELST)
                                              (FLENGTH LISPXTEM)))
                              (PRINT '|archived| T T))
                  (T (PRINT '(|no| |archive| |list|)
```

```
                              T))))
            (GO NOTCOM))
        (RETURN '\`)
    NOTCOM
        (COND
           ((SETQ LISPY (GETPROP LISPXOP '*HISTORY*))          (* |command| |defined| |by| \a NAME |command.|)
            (COND
               ((NULL LISPXLINE)
                (COND
                   ((AND (OR (EQ LISPXID '_)
                             (EQ LISPXID '\:))
                         (BOUNDP LISPXOP))
```

(* |User| |typed| |command| |followd| |by| |just| |c.r.| |since| |command| |is| |also| |the| |name| |of| \a |variable,| |thats| |probably| |what| |he| |wants,| |especially| |since| |he| |can| |always| |say| REDO @ FOO)

```
                    (SETQ LISPY NIL))
                   (T (GO REDOCOM))))
               ((NULL (CAR LISPY))
                (ERROR LISPXOP '"doesn't take any arguments"))
               (T (GO REDOCOM)))
            (SETQ LISPY NIL))
           ((FMEMB LISPXOP LISPXCOMS)
```

(* |Since| LISPXOP |is| |not| |one| |of| |the| |built| |in| |commands,| |and| |not| |on| LISPXMACROS\, |presumably| |the| |user| |has| |included| |it| |on| LISPXCOMS |because| |he| |is| |going| |to| |process| |it| |in| LISPXUSERFN.
I\n |any| |event,| |dont| |want| |to| |do| |any| |spelling| |correction.|)

```
            (AND LISPXLISTFLG (SETQ LISPXLINE NIL))
            (GO DO-IT)))
        (COND
           (LISPXLISTFLG                                       (* |Input| |is| \a |single| |list.|)
                  (COND
                     ((EQ (CAR LISPXX)
                          'LAMBDA)
                      (SETQ LISPXLINE (LIST (LISPXREAD T T))))
                     (T (AND (LITATOM (CAR LISPXX))
                             (COND
                                ((OR (FGETD (CAR LISPXX))
                                     (GETLIS (CAR LISPXX)
                                             MACROPROPS)
                                     (GETLIS (CAR LISPXX)
                                             '(EXPR FILEDEF CLISPWORD)))
                                 (AND ADDSPELLFLG (ADDSPELL (CAR LISPXX)
                                                            2)))
                                ((AND DWIMFLG (SETQ LISPXOP (FIXSPELL (CAR LISPXX)
                                                                      70 LISPXCOMS NIL LISPXX)))
                                 (SETQ LISPXLINE (CDR LISPXX))
                                 (GO SELECT))))
                        (AND LISPXLISTFLG (SETQ LISPXLINE NIL))))
                (GO DO-IT))
           ((NULL LISPXLINE)                                   (* |Input| |is| \a |single| |atom.|)
            (AND (LITATOM LISPXX)
                 (COND
                    ((BOUNDP LISPXX)
                     (AND ADDSPELLFLG (ADDSPELL LISPXX 3)))
                    ((AND DWIMFLG (SETQ LISPXOP (FIXSPELL LISPXX 70 LISPXCOMS NIL T)))
                     (COND
                        ((LISTP LISPXOP)                       (* RUN-ON |spelling| |error.|)
                         (SETQ LISPXLINE (LIST (CDR LISPXOP)))
                         (SETQ LISPXOP (COND
                                          ((LISTP (CAR LISPXOP))
                                                               (* |synonym|)
                                           (CADAR LISPXOP))
                                          (T (CAR LISPXOP))))))
                     (SETQ LISPXX LISPXOP)
                     (GO SELECT))))
            (GO DO-IT))
           ((NOT (LITATOM LISPXX)))
           ((FGETD LISPXX)
```

(* |put| |on| SPELLINGS2 |even| |though| |in| APPLY |format| |since| |is| |also| |good| |in| EVAL |format|)

```
            (AND ADDSPELLFLG (ADDSPELL LISPXX 2)))
           ((AND DWIMFLG (NULL (GETLIS LISPXX '(EXPR FILEDEF)))
                 (SETQ LISPXOP (FIXSPELL LISPXX 70 LISPXCOMS NIL T)))
            (COND
               ((LISTP LISPXOP)
                (SETQ LISPXLINE (CONS (CDR LISPXOP)
                                      LISPXLINE))
                (SETQ LISPXOP (CAR LISPXOP))))
            (SETQ LISPXX LISPXOP)
            (GO SELECT)))
    DO-IT
        (AND (NULL DONTSAVEFLG)
             (SETQ LISPXHIST (HISTORYSAVE LISPXHISTORY LISPXID NIL LISPXX LISPXLINE)))
        (COND
```

```
            (LISPY (SETQ LISPXVALUE (CAR (SETQ LISPXVALUES (CL:MULTIPLE-VALUE-LIST
                                                             (LET ((LISPXLINE (COND
                                                                                (LISPXLISTFLG (CDR LISPXX))
                                                                                (T (NLAMBDA.ARGS LISPXLINE)))))
                                                                   (EVAL (OR (CADR LISPY)
                                                                             (CADDR LISPY))
                                                                         LISPXID)))))))
            ((AND LISPXXUSERFN (CL:FUNCALL LISPXXUSERFN LISPXX LISPXLINE))
             (COND
                (LISPXVALUES (SETQ LISPXVALUE (CAR LISPXVALUES)))
                (T (SETQ LISPXVALUES (LIST LISPXVALUE)))))
            (T (SETQ LISPXVALUE (CAR (SETQ LISPXVALUES (CL:MULTIPLE-VALUE-LIST
                                                         (COND
                                                            ((NULL LISPXLINE)
                                                                (* A |form.|)
                                                             (EVAL (COND
                                                                      ((NLISTP LISPXX)
                                                                       LISPXX)
                                                                      (T (LISPX/ LISPXX)))
                                                                   LISPXID))
                                                            ((OR (CDR LISPXLINE)
                                                                 (AND CLISPFLG (LITATOM LISPXX)
                                                                      (CAR LISPXLINE)
                                                                      (LITATOM (CAR LISPXLINE))
                                                                      (NEQ (SETQ LISPXTEM (NTHCHAR (CAR
                                                                                                     LISPXLINE
                                                                                                     )
                                                                                                   1))
                                                                           '-)
                                                                      (FMEMB LISPXTEM CLISPCHARS)
                                                                      (NEQ (ARGTYPE LISPXX)
                                                                           3)))
                                                                (* |The| |special| |checks| |are| |to| |enable| |constructs| |like| FOO _T |to| |work,| |even| |when| FOO |is| |also| |the| |name|
                                                                |of| \a |function,| |i.e.| |instead| |of| |applying| FOO |to| _T\, (|which| |would| |cause| |an| |unusual| CDR ARGLIST |error|)
                                                                (FOO _ T) |is| |evaluated,| |which| |will| |invoke| DWIM.)
                                                             (COND
                                                                ((NEQ (ARGTYPE LISPXX)
                                                                      3)
                                                                 (PRIN1 " = " T)
                                                                 (PRINT (CONS LISPXX LISPXLINE)
                                                                        T)))
                                                             (EVAL (LISPX/ (CONS LISPXX LISPXLINE))
                                                                   LISPXID))
                                                            (T (APPLY (LISPX/ LISPXX)
                                                                      (LISPX/ (CAR LISPXLINE)
                                                                              LISPXX)
                                                                      LISPXID)))))))))
            (AND LISPXHIST (LISPXSTOREVALUE LISPXHIST LISPXVALUE LISPXVALUES))
            (RETURN (PROGN (SETQ IT LISPXVALUE)
                           (|for| X |in| LISPXVALUES |do| (SHOWPRINT X T T))
                           (CL:VALUES-LIST LISPXVALUES)))
REDOCOM
            (SETQ LISPXX (COND
                            (LISPXLISTFLG (LIST LISPXX))
                            (T (CONS LISPXX LISPXLINE))))          (* |The| |entire| |history| |command.|)
            (AND (NULL DONTSAVEFLG)
                 (SETQ LISPXHIST (HISTORYSAVE LISPXHISTORY LISPXID NIL NIL NIL (LIST '*HISTORY* LISPXX
                                                                                     '*GROUP* NIL))))
            (SELECTQ LISPXOP
                 (ORIGINAL (SETQ LISPY (APPEND LISPXLINE)))
                 (Ÿ (SETQ LISPY (LISPXUSEC LISPXLINE LISPXHISTORY)))
                 ((|retry| RETRY)
                    (SETQ LISPY (CONS 'RETRY\: (APPEND (LISPXFIND LISPXHISTORY LISPXLINE 'INPUT T)))))
                 ((|name| NAME)
                    (SETQ LISPXTEM (CDR (OR (SETQ LISPZ (OR (FMEMB '\: LISPXLINE)
                                                            (FMEMB 'IN LISPXLINE)
                                                            (FMEMB '|in| LISPXLINE)))
                                            LISPXLINE)))
                    (* LISPXTEM |coresponds| |to| |the| |event| |specification,| LISPZ |to| |the| |end| |of| |the| |arguments,| |if| |any.|)
                    (SETQ LISPZ (COND
                                   ((NULL LISPZ)
                                    NIL)
                                   ((CDR (SETQ LISPZ (LDIFF (CDR LISPXLINE)
                                                            LISPZ)))
                                    LISPZ)
                                   ((LISTP (CAR LISPZ))          (* |user| |got| |confused| |and| |put| |in| |an| |extra| |set| |of|
                                                                 |parens.|)
                                    (CAR LISPZ))
                                   (T LISPZ)))
                    (SETQ LISPY (LISPXFIND LISPXHISTORY LISPXTEM 'INPUT T))
                    (RESETVARS ((EDITQUIETFLG T))
                               (MAPC LISPZ (FUNCTION (LAMBDA (X)
                                                       (COND
```

```
                                                    ((NOT (HISTORYMATCH LISPY (EDITFPAT X T)))
                                                     (LISPXPRIN1 X T)
                                                     (MAPRINT LISPXTEM T '" does not appear in "
                                                            '\
NIL NIL T)))))))
                    (/PUT (CAR LISPXLINE)
                          '*HISTORY*
                          (CONS LISPZ (CONS (APPEND LISPY)
                                            (LISPXFIND LISPXHISTORY LISPXTEM 'COPIES T))))

          (* |The| |reason| |for| |storing| |the| |input| |separate| |frm| |the| |event|
          (\s) |is| |that| |the| |user| |may| |have| |performed| NAME FOO USE -
          |meaning| |the| USE |input,| |rather| |than| |the| |normal| |input.| |The| |reason| |for| |the| |append| |is| |that| |lispy| |will| |also|
          |be| |the| |input| |portion| |of| |the| |name| |event| |on| |the| |history| |list,| |and| |we| |want| |it| |not| |to| |be| |smashed| |when|
          |that| |entry| |is| |slips| |off| |the| |end| |of| |the| |history| |list.|)

                    (/REMPROP (CAR LISPXLINE)
                              'STATE)
                    (/SETATOMVAL 'LISPXCOMS (UNION (LIST (CAR LISPXLINE))
                                                   LISPXCOMS))
                    (/SETATOMVAL 'HISTORYCOMS (UNION (LIST (CAR LISPXLINE))
                                                     HISTORYCOMS))
                    (COND
                       ((GETD (CAR LISPXLINE))
                        (MAPRINT (CONS (CAR LISPXLINE)
                                       '(|is| |also| |the| |name| |of| \a |function.| |When| |typed| |in,| |its|
                                         |interpretation| |as| \a |history| |command| |will| |take|
                                         |precedence.|))
                                 T "****Note: " '\
NIL NIL T)))
                    (PRINT (CAR LISPXLINE)
                           T T))
                ((REDO |redo| REPEAT |repeat|)
                    (COND
                       ((NULL (SOME LISPXLINE (FUNCTION (LAMBDA (X TAIL)
                                                   (SELECTQ (CAR TAIL)
                                                        ((WHILE UNTIL |while| |until|)
                                                            (COND
                                                               ((AND (CDR TAIL)
                                                                     (NEQ (CAR (SETQ LISPXTEM
                                                                                     (NLEFT LISPXLINE 1 TAIL)))
                                                                          'F))
                                                                (* |backs| |up| |one|)
                                                                (SETQ LISPXLINE (AND LISPXTEM
                                                                                     (LDIFF LISPXLINE
                                                                                            (CDR LISPXTEM)
                                                                                            )))
                                                                (AND (NULL (CDDR (SETQ LISPXTEM
                                                                                       (CDR TAIL))))
                                                                     (OR (LISTP (CAR LISPXTEM))
                                                                         (BOUNDP (CAR LISPXTEM))
                                                                         (NOT (FNCHECK (CAR LISPXTEM)
                                                                                       T T T LISPXTEM)))
                                                                     (SETQ LISPXTEM (CAR LISPXTEM)))
                                                                (COND
                                                                   ((OR (EQ (CAR TAIL)
                                                                            'UNTIL)
                                                                        (EQ (CAR TAIL)
                                                                            '|until|))
                                                                    (SETQ LISPXTEM (LIST 'NOT LISPXTEM))))
                                                                T)))
                                                        ((TIMES |times|)
                                                            (COND
                                                               ((AND (NULL (CDR TAIL))
                                                                     (SETQ LISPXTEM (NLEFT LISPXLINE 1
                                                                                           TAIL))
                                                                     (NEQ (CAR LISPXTEM)
                                                                          'F))
                                                                (SETQ LISPXLINE (LDIFF LISPXLINE LISPXTEM)
                                                                 )
                                                                (SETQ LISPXTEM (OR (NUMBERP (CAR LISPXTEM)
                                                                                            )
                                                                                   T)))))
                                                        NIL)))))
                            (SETQ LISPXTEM (OR (EQ LISPXOP 'REPEAT)
                                               (EQ LISPXOP '|repeat|)))))
                    (SETQ LISPY (LISPXFIND LISPXHISTORY LISPXLINE 'INPUT T))
                    (COND
                       ((EQ LISPXID '*)                          (* |For| |editor.|)
                        (SETQ LISPY (COPY LISPY)))
                       (T                                        (* |Cant| |allow| |same| |input| |to| |appear| |twice| |in| |history.|)
                          (SETQ LISPY (APPEND LISPY))))
                    (COND
                       (LISPXTEM (SETQ LISPY (LIST HISTSTR2 LISPXTEM LISPY)))))
                ((FIX |fix|)
                    (SETQ LISPY (COPY (LISPXFIND LISPXHISTORY (COND
                                                         ((SETQ LISPXTEM (FMEMB '- LISPXLINE))
```

(* |User| |can| |say| FIX -
|and| |give| |the| |commands.| |Then| |he| |doesn't| |have| |to| |wait| |for| |editor| |to| |print| EDIT\, |and| |him| |to| |type| OK |at| |the| |end.| |Also,| |the| |commands| |stored| |on| |the| |history| |list| |in| |this| |fashion| |can| |be| |reexecuted| |by| \a REDO FIX |command.|)

```
                                                          (LDIFF LISPXLINE LISPXTEM))
                                                        (T LISPXLINE))
                                    'INPUT T)))
          (SETQ LISPY (COND
                         ((STREAMPROP (GETSTREAM T)
                                      'FIXFN)
                          (APPLY* (STREAMPROP (GETSTREAM T)
                                              'FIXFN)
                                  (GETSTREAM T)
                                  LISPY
                                  (CDR LISPXTEM)))
                         (T (LISPXFIX LISPY (CDR LISPXTEM)))))
```

(* |usually| |defined| |as| |just| \a |call| |to| EDITL |but| |can| |be| |advised| |to| |handle| |string| |situations,| |such| |as| |in| BARD. I\f |the| |stream| |has| \a FIX |function| APPLY |it| |instead| |of| |the| |default|)

```
      )
     ((USE |use|)
          (SETQ LISPY (LISPXUSE LISPXLINE LISPXHISTORY LISPXHIST)))
     (|...| (COND
               ((NULL LISPXLINE)
                (ERROR '"... what??" '\' T)))
            (SETQ LISPY (LISPXFIND LISPXHISTORY NIL 'ENTRY T))
            (SETQ LISPXTEM (COND
                                ((LISTGET1 LISPY '...ARGS))
                                ((SETQ LISPXTEM (LISTGET1 LISPY 'USE-ARGS))
```

(* |The| CAAAR |is| |because| CAR |is| |the| |list| |of| USEARGS |which| |is| \a |list| |of| |list| |of| |variables.|)

```
                                 (CONS (CAAAR LISPXTEM)
                                       (CDR LISPXTEM)))
                                ((SETQ LISPXTEM (LISTGET1 LISPY '*HISTORY*))
                                                          (* E.\g. \a |lispxmacro| |or| |lispxhistorymacro.|)
                                 (CONS (CADR LISPXTEM)
                                       (LISPXGETINPUT LISPXTEM (CONS LISPXTEM (CDR LISPY)))))
                                (T (SETQ LISPY (LISPXFIND LISPXHISTORY NIL 'INPUT T))
                                   (CONS (COND
                                            ((OR (NULL (CDR LISPY))
                                                 (EQ (CADR LISPY)
                                                     HISTSTR0))
                                                          (* EVAL |input,| |substitute| |for| |first| |argument| |which| |is|
                                                          CADAR)
                                             (CADAR LISPY))
                                            ((NLISTP (CADR LISPY))
                                                          (* |e.g.| PP FOO)
                                             (CADR LISPY))
                                            (T            (* APPLY |input.| |e.g.| LOAD (FOO) |substitute| |for| FOO)
                                               (CAADR LISPY)))
                                         LISPY))))        (* LIPXTEM |is| |now| \a |dotted| |pair| |of| |aagument| |and|
                                                          |input.|)
            (NCONC LISPXHIST (LIST '...ARGS LISPXTEM))
            (SETQ LISPY (LISPXUSE0 (LIST LISPXLINE)
                                (LIST (LIST (CAR LISPXTEM)))
                                (LIST (CDR LISPXTEM)))))
     (SETQ LISPY (COND
                    ((EQ (CAR LISPY)
                         LISPXOP)                     (* |from| |lispxhistorymacro.|)
                     (EVAL (OR (CADR LISPY)
                               (CADDR LISPY))
                           LISPXID))
                    ((NULL (CAR LISPY))               (* |Command| |defined| |by| |name| |command,| |with| |no|
                                                      |arguments|)
                     (APPEND (CADR LISPY)))
                    (T                                (* |From| |name| |command.|)
                       (LISPXUSE0 (LIST LISPXLINE)
                              (LIST (CAR LISPY))
                              (LIST (CADR LISPY)))))))) (* LISPY |is| |now| |the| |input.|)
(AND (NULL REREADFLG)
     (FMEMB HISTSTR2 (LISTP LISPY))
     (SETQ REDOCNT -1))
```

(* |the| -1 |is| |because| |the| |first| |thing| |that| |will| |happen| |will| |be| \a |call| |to| |lispxrepeatread| |which| |will| |increment| |redocnt| |to| 0 |the| |check| |is| |made| |here| |instead| |of| |inside| |the| |selectq| |at| REDO |because| |of| |cases| |where| |user| |does| USE |on| |an| |event| |involving| \aREPEAT |input|)

```
(AND LISPXHIST (FRPLACA LISPXHIST LISPY))
(COND
   ((EQ LISPXOP 'NAME)
```

(* NAME |is| |handled| |as| \a |history| |command| |so| |that| |the| |command| |is| |stored| |before| |it| |tries| |to| |do| |the| |lookup,| |and| |to| |share| |in| |other| |common| |code.| |but| |it| |is| |not| |actually| |redone| |or| |unread.|)

```
            )
           (LISPXFLG (RESETVARS (READBUF)
                                (LISPXUNREAD LISPY LISPXHIST)
                         LP  (COND
                                ((NULL (SETQ READBUF (LISPXREADBUF READBUF)))
                                 (RETURN)))
                                (LISPX (LISPXREAD T T)
                                       LISPXID)
                                (GO LP)))
              (T (LISPXUNREAD LISPY LISPXHIST)))
         (RETURN LISPXHIST))))
```

(LISPX/

```
(LAMBDA (X FN VARS)                                    (* |lmm| "16-FEB-83 06:42")
  (COND
     ((OR (NULL LISPXFNS)
          (NULL LISPXHISTORY))
      X)
     (FN

        (* I\f FN |is| |not| NIL\, |it| |is| |the| |name| |of| \a |function| |and| X |is| |its| |argument| |list.|
        |Subsitution| |only| |occurs| |for| |functions| |that| EVAL\, |such| |as| RPAQ\, SETQ\, |and| \e.)

         (COND
            ((NLISTP X)                                (* X |is| |an| (|atomic|) |argument| |list,| |e.g.|
                                                       |type| PP FOO\, |don't| |substitute| |for| FOO.)
             X)
            ((SELECTQ (ARGTYPE FN)
                  ((1 3)

        (* |Slightly| |different| |check| |than| |in| LISPX/1 |and| DWIMIFY1\, |etc.|
        |This| |check| |wants| |to| |know| |whether| |this| |function| |calls| |eval| |explicitly| |itself.|
        |The| |others| |say| |are| |the| |aaguments| |evaluated| |either| |by| |virtue| |of| |it| |being| \a |normal| |function,| |or| |an| |eval|
        |call.|)

                    (EQMEMB 'EVAL (GETPROP FN 'INFO)))
                NIL)
             (LISPX/1 X T))
            (T X)))
     ((LISTP X)                                        (* X |is| \a |form.|)
      (LISPX/1 X))
     (T (OR (CDR (FASSOC X LISPXFNS))
            X)))))
```

(LISPX/1

```
(LAMBDA (X TAILFLG)                                    (* |lmm| " 2-Jul-85 02:20")
  (AND X (PROG ((TEM1 (CAR X))
                TEM2 TEM3)
               (COND
                  ((NLISTP X)
                   (RETURN X))
                  ((LISTP (CAR X))
                   (SETQ TEM1 (LISPX/1 (CAR X)))
                   (GO DO-CDR)))
               (COND
                  (TAILFLG (GO DO-CDR)))
               (SETQ TEM1 (OR (CDR (FASSOC (CAR X)
                                           LISPXFNS))
                              (CAR X)))
               (SELECTQ (CAR X)
                    (QUOTE (RETURN X))
                    ((FUNCTION F/L)
                         (SETQ TEM2 (LISPX/1 (CDR X)))
                         (GO DO-CDR1))
                    ((LAMBDA NLAMBDA)
                         (SETQ TEM3 (CADR X))
                         (PROG ((VARS (COND
                                          ((NLISTP TEM3)
                                           (CONS TEM3 VARS))
                                          (T (APPEND TEM3 VARS)))))
                               (SETQ TEM2 (LISPX/1 (CDDR X)
                                                   T)))
                         (GO DO-CDDR1))
                    (PROG (PROG ((VARS (NCONC (MAPCAR (CADR X)
                                                      (FUNCTION (LAMBDA (X)
                                                                  (COND
                                                                     ((ATOM X)
                                                                      X)
                                                                     (T (SETQ TEM3 T)
                                                                        (CAR X))))))
                                              VARS)))
                                (SETQ TEM2 (LISPX/1 (CDDR X)
                                                    (CAR X))))
                          (COND
```

```
                            ((NULL TEM3)
                             (GO DO-CDDR1)))
                           (RETURN (CONS 'PROG (CONS (MAPCAR (CADR X)
                                                             (FUNCTION (LAMBDA (X)
                                                                       (COND
                                                                          ((ATOM X)
                                                                           X)
                                                                          (T (LISPX/1 X T))))))
                                                     TEM2))))
                     (SETQ (COND
                              ((FMEMB (CADR X)
                                      VARS)                    (* |don't| |have| |to| |be| |undoable| |for| |bound| |vriabes,| |e.g.|
                                                               |in| MAPC\, PROG\, |etc.|)
                               (SETQ TEM1 (CAR X))
                               (GO DO-CDDR))))
                     (COND
                        ((AND (OR (EQ (SETQ TEM2 (ARGTYPE (CAR X)))
                                      1)
                                  (EQ TEM2 3))
                              (NOT (OR (EQ (SETQ TEM2 (GETPROP (CAR X)
                                                               'INFO))
                                           'EVAL)
                                       (FMEMB 'EVAL TEM2))))
```

(* D\o |not| |substitute| |unless| |you| |know| |that| |the| |function| |will| |evaluate| |its| |arguments,| |as| |with| ERSETQ\, RESETVAR\, |etc.| |The| |eason| |for| |not| |just| |returning| |is| |that| |the| |function| |name| |may| |be| |on| |lispxfns,| |e.g.| SETQQ |becomes| SAVESETQQ.)

```
                      (SETQ TEM2 (CDR X))
                      (GO DO-CDR1))
                     ((AND CLISPARRAY (NULL (FGETD (CAR X)))
                           (SETQ TEM3 (GETHASH X CLISPARRAY)))
                      (RETURN (LISPX/1 TEM3)))
                     ((NULL (FGETD (CAR X)))
```

(* |lispx/| |will| |get| |caaled| |again| |anyway| |after| |it| |is| |translated,| |and| |if| |we| |do| |substitution| |now,| |may| |change| \a |setq| |to| SAVESETQ |that| |refers| |to| \a |variable| |bound| |in| \a BIND\, |etc.|)

```
                      (RETURN X))))
                DO-CDR
                     (SETQ TEM2 (LISPX/1 (CDR X)
                                         T))
                DO-CDR1
                     (RETURN (COND
                                ((AND (EQ TEM1 (CAR X))
                                      (EQ TEM2 (CDR X)))
                                 X)
                                (T (CONS TEM1 TEM2))))
                DO-CDDR
                     (SETQ TEM2 (LISPX/1 (CDDR X)
                                         T))
                DO-CDDR1
                     (RETURN (COND
                                ((AND (EQ TEM1 (CAR X))
                                      (EQ TEM2 (CDDR X)))
                                 X)
                                (T (CONS TEM1 (CONS (CADR X)
                                                    TEM2)))))))))
```

(LISPXEVAL

```
(LAMBDA (LISPXFORM LISPXID)
```

(* |Evaluates| LISPXFORM |same| |as| |though| |were| |typed| |in| |to| LISPX.
I\f LISPXID |not| |given,| _ |is| |used.|)

```
  (PROG (LISPXHIST)
        (OR LISPXID (SETQ LISPXID '_))
        (SETQ LISPXHIST (HISTORYSAVE LISPXHISTORY LISPXID NIL LISPXFORM))
        (FRPLACA (CDDR LISPXHIST)
               (EVAL (COND
                        ((NLISTP LISPXFORM)
                         LISPXFORM)
                        (T (LISPX/ LISPXFORM)))
                     LISPXID))
        (RETURN (CADDR LISPXHIST)))))
```

(LISPXSTOREVALUE

```
(LAMBDA (EVENT VALUE VALUES)                                   (* |lmm| " 1-May-86 12:36")
  (COND
     (EVENT (FRPLACA (CDDR EVENT)
                     VALUE)
            (LISPXPUT 'LISPXVALUES VALUES NIL EVENT)))))
```

(HISTORYSAVE

```
(LAMBDA (HISTORY ID INPUT1 INPUT2 INPUT3 PROPS)                    (* |wt:| "18-NOV-78 21:52")

        (* HISTORY |is| |of| |the| |form| (LIST INDEX SIZE MOD) INDEX |is| |between| 0 |and| MOD
        (MOD |is| |usually| 100 |or| \a |multiple| |of| 100) |and| |is| |automatically| |incremented| |each| |time| |an| |entry| |is| |added.|
        SIZE |is| |the| |length| |of| LIST\, |and| |after| LIST |reaches| |that| |length,| |old| |entries| |at| |the| |end| |are| |cannibalized|
        |and| |moved| |to| |the| |front| |when| |new| |entries| |are| |added.|
        |The| |form| |of| |each| |entry| |on| |the| HISTORY |list| |is| (INPUT ID VALUE . PROPS) |Value| |is| |initialized| |to| \.)

        (* |the| |value| |of| |historysave| |is| |the| |corresponding| |event| |or| |subevent| |in| |the| |case| |of| |gruped| |events.|
        |Groups| |are| |represented| |by| |the| |value| |of| |the| |property| *GROUP* |which| |is| \a |list| |of| |the| |form|
        (|event| |event| |...| |event|)\. |each| |subevent| |can| |have| |its| |own| *GROUP* |property,| |or| HISTORY |property,| |etc.|
        HISTORYSAVE |automatically| |retrieves| |the| |appropraite| |subevetn,| |no| |matter| |ho| |nested,| |when| |given| |an| |input|
        |that| |has| |been| |reread,| |so| |the| |calling| |functio| |doesnt| |hae| |to| |distinguish| |between| |new| |input| |and|
        |reexecution| |of| |input| |whose| |history| |entry| |has| |alredy| |been| |set| |up.|)

  (PROG ((L (CAR HISTORY))
         (INDEX (CADR HISTORY))
         (SIZE (CADDR HISTORY))
         (MOD (OR (CADDDR HISTORY)
                  100))
         (N 0)
         X Y TEM)
        (COND
           ((OR (NLISTP HISTORY)
                (AND (NLISTP (CAR HISTORY))
                     (CAR HISTORY)))
            (RETURN NIL))
           ((AND REREADFLG (SETQ X (CDR (FMEMB '*GROUP* (CADR (FMEMB HISTSTR3 REREADFLG))))))

        (* |This| |input| |is| |the| |result| |of| \a |history| |command,| |so| |do| |not| |make| \a |new| |entry.|)

            (COND
               ((AND (FMEMB HISTSTR2 REREADFLG)
                     (NOT (ILESSP REDOCNT \#REDOCNT)))
                (COND
                   ((SETQ TEM (CDR (FMEMB '*REDOCNT* (SETQ X (CAAR HISTORY)))))
                    (FRPLACA TEM REDOCNT))
                   (T (NCONC X (LIST '*REDOCNT* REDOCNT))))
                (RETURN X)))
            (FRPLACA X (NCONC1 (CAR X)
                               (SETQ Y (CONS (COND
                                                (INPUT1 (CONS INPUT1 (CONS INPUT2 INPUT3)))
                                                (INPUT2 (CONS INPUT2 INPUT3))
                                                (T INPUT3))
                                             (CONS ID (CONS '\` PROPS))))))
            (RETURN Y)))
        (COND
           ((IGREATERP (SETQ INDEX (ADD1 INDEX))
                   MOD)
            (SETQ INDEX (IPLUS INDEX (MINUS MOD)))))
    LP  (COND
           ((CDDR L)
            (ADD1VAR N)
            (SETQ L (CDR L))
            (GO LP))
           ((IGREATERP SIZE (IPLUS N 2))
            (FRPLACA HISTORY (CONS (SETQ X (LIST NIL NIL NIL))
                                   (CAR HISTORY)))
            (GO SMASH)))
        (SETQ X (CDR L))
        (COND
           ((AND ARCHIVELST (NEQ HISTORY EDITHISTORY)
                 (OR (AND ARCHIVEFN (ARCHIVEFN (CAAR X)
                                               (CAR X)))
                     (LISTGET1 (CAR X)
                            '*ARCHIVE*)))
            (FRPLACA ARCHIVELST (CONS (LISPXFIND1 (CAR X))
                                      (CAR ARCHIVELST)))
            (FRPLACA (CDR ARCHIVELST)
                   (ADD1 (CADR ARCHIVELST)))))
        (FRPLACD L NIL)                                            (* |Moves| |last| |entry| |to| |front.|)
        (FRPLACA HISTORY (FRPLACD X (CAR HISTORY)))
        (SETQ X (CAR X))                                           (* X |is| |now| |the| |entry| |to| |be| |canniablized.|)
    SMASH
        (FRPLACA (CDR HISTORY)
               INDEX)
        (COND
           ((LISTP ID)                                             (* ID |is| |the| |new| |entry.|)
            (FRPLACA (CAR HISTORY)
                     (SETQ Y ID))
            (GO OUT))
           ((NLISTP (SETQ Y (CAR X)))                              (* Y |is| |now| |the| |input| |portion| |of| |the| |entry.|)
            (SETQ Y (CONS NIL NIL))))
        (COND
           (INPUT1

        (* |Means| INPUT |is| (INPUT1 INPUT2 . INPUT3) |used| |primarily| |for| APPLY INPUT |when| INPUT1 |is| |function| |and|
        INPUT2 |args.|)
```

```
                  (COND
                     ((CDR Y)                                          (* |Cannibalize| |previous| |input.|)
                      (FRPLACA Y INPUT1)
                      (FRPLACA (CDR Y)
                               INPUT2)
                      (FRPLACD (CDR Y)
                               INPUT3))
                     (T (SETQ Y (CONS INPUT1 (CONS INPUT2 INPUT3))))))
               (INPUT2                                                 (* |Means| INPUT |is| (INPUT2 . INPUT3) |used| |primarily| |for|
                                                                       EVAL INPUT |when| INPUT2 |is| |form.|)
                      (FRPLACA Y INPUT2)
                      (FRPLACD Y INPUT3))
               (T                                                      (* |Means| INPUT |is| INPUT3\, |used| |primarily| |for| |line|
                                                                       |inputs,| |such| |as| HISTORY |commands.|)
                  (SETQ Y INPUT3)))
            (FRPLACA X Y)
            (FRPLACA (SETQ Y (CDR X))
                     ID)
            (COND
               ((EQ (CADR Y)
                    '\`)

            (* Y |may| |correspond| |to| |an| |event| |that| |has| |not| |yet| |completed| |but| |will,| |e.g.|
            |you| |are| |in| \a |break| |and| |have| |performed| |more| |than| 30 |operations.|
            |Therefore| Y\, |or| |at| |least| |that| |part| |of| Y |beginning| |with| |the| |value| |field,| |should| |not| |be| |used| |since| |it| |will|
            |be| |smashed| |wen| |the| |event| |finishes.|)

                (FRPLACD Y (SETQ Y (CONS '\` PROPS))))
               (T (FRPLACA (SETQ Y (CDR Y))
                           '\`)
                  (FRPLACD Y PROPS)))
            (COND
               (HISTORYSAVEFORMS (PROG ((EVENT X))
                                       (MAPC HISTORYSAVEFORMS (FUNCTION (LAMBDA (X)
                                                                        (ERSETQ (EVAL X))))))))
         OUT (COND
               ((EQ ID '*)
                (LISPXWATCH EDITSTATS))
               (T (LISPXWATCH LISPXSTATS)))
            (COND
               ((EQ REREADFLG 'ABORT)
                (ERROR!)))
            (RETURN X))))
```

(LISPXFIND

```
(LAMBDA (HISTORY LINE TYPE BACKUP QUIETFLG)                            (* |wt:| 24-JUN-76 14 18)

        (* QUIETFLG=T |means| |tell| |editor| |not| |to| |print| |messages| |on| |alt-mode| |matches.|
        |Used| |by| LISPXUSE |and| LISPXUSEC.)

   (COND
      ((NULL HISTORY)
       (ERROR '"no history." '\` T)))

        (* LINE |specifies| |an| |entry| |or| |entries| |on| HISTORY\, |and| TYPE |the| |desired| |format| |of| |the| |value.|
        LISPXFIND |uses| HISTORYFIND |to| |get| |the| |corresponding| |entries,| |and| |then| |decides| |what| |to| |do| |with| |them.|)

   (RESETVARS ((EDITQUIETFLG (OR EDITQUIETFLG QUIETFLG)))
              (RETURN (PROG ((LST (CAR HISTORY))
                             (INDEX (CADR HISTORY))
                             (MOD (OR (CADDDR HISTORY)
                                      100))
                             (LINE0 LINE)
                             VAL TEM)
                            (COND
                               (BACKUP

        (* |Used| |when| |want| |to| |refer| |to| HISTORY |before| |last| |entry| |was| |made,| |e.g.|
        |for| UNDO |so| UNDO UNDO |will| |work.|)

                                       (SETQ LST (CDR LST))
                                       (SETQ INDEX (SUB1 INDEX))))
                               (COND
                                  ((AND REREADFLG (NULL (CAAR LST)))

        (* |Special| |glitch| |to| |allow| \a |bad| |history| |command| |which| |contains| |relative| |event| |numbers| |to| |be| |reexecuted|
        |without| |changing| |the| |event| |specification| |provided| |it| |is| |done| |immediately,| |e.g.|
        |user| |types| USE FOO FOR FIE IN -2\, LISPX |types| FIE ? |user| |can| TYPE USE FUM FOR FIE\, |and| -2 |will| |refer| |to|
        |the| |correct| |event.|)

                                        (SETQ LST (CDR LST))
                                        (SETQ INDEX (SUB1 INDEX))))
                                FIND
                                (COND
                                   ((NULL LINE)
```

```
                        (SETQ VAL (CAR LST))
                        (COND
                           ((AND (OR (EQ (CAAR VAL)
                                         'UNDO)
                                     (EQ (CAAR VAL)
                                         '|undo|))
                                 (NEQ (CADDR VAL)
                                      '\`))                    (* S\o |can| |say| UNDO |then| REDO |or| USE.)
                            (SETQ VAL (CADR LST))))
                        (GO SINGLE))
                       ((EQ (CAR LINE)
                            '@@)                               (* |Archive.|)
                        (RETURN (LISPXFIND ARCHIVELST (CDR LINE)
                                           TYPE))))
                LP  (SETQ VAL (NCONC VAL (LISPXFIND0 (LDIFF LINE0 (SETQ LINE0
                                                                         (OR (FMEMB 'AND (CDR LINE0))
                                                                             (FMEMB '|and| (CDR LINE0)))))
                                                     LST INDEX MOD)))
                    (COND
                       ((SETQ LINE0 (CDR LINE0))
                        (GO LP)))
                GROUP
                    (COND
                       ((NULL (CDR VAL))
                        (SETQ VAL (CAR VAL))
                        (GO SINGLE)))                          (* VAL |is| \a |list| |of| |events.|)
                    (AND ARCHIVEFLG (MAPC VAL (FUNCTION (LAMBDA (X)
                                                          (LISPXPUT '*ARCHIVE* T NIL X)))))
                    (RETURN (AND VAL (SELECTQ TYPE
                                              (INPUT (MAPCONC VAL
                                                              (FUNCTION (LAMBDA (VAL)
                                                                          (APPEND
                                                                           (SETQ TEM
                                                                            (LISPXGETINPUT
                                                                             (COND
                                                                                ((NULL (CAR VAL))
                                                                                 (LISTGET1 VAL '*HISTORY*))
                                                                                (T (CAR VAL)))
                                                                             VAL))
                                                                           (AND (NEQ (CAR (LAST TEM))
                                                                                     HISTSTR0)
                                                                                (LIST HISTSTR0)))))))
                                              ((ENTRY ENTRIES)
                                                  VAL)
                                              ((COPY COPIES)
                                                  (MAPCAR VAL (FUNCTION LISPXFIND1)))
                                              (GO BAD))))
```

(* |For| COPIES |and| ENTRIES\, |calling| |function| |expects| \a LIST |of| |events,| |for| COPY |and| ENTRY |only| |one.| (|however| |if| |the| |event| |specification| |produces| |more| |than| |one| |event,| LISPXFIND |treats| COPY |and| ENTRY |the| |same| |as| COPIES |and| ENTRIES.) ENTRY |is| |used| |by| LISPXUSE |and| |the| |...| |Command.| |Entries| |is| |used| |by| FORGET. COPIES |is| |used| |by| NAME |and| ARCHIVE.
-
REDO |is| |the| |same| |as| INPUT |except| |that| |the| |value| |returned| |will| |not| |be| |copied| |again,| |so| |it| |must| |be| |copied| |here.|)

```
                SINGLE
                                                           (* VAL |is| \a |single| |event.|)
                    (AND ARCHIVEFLG (LISPXPUT '*ARCHIVE* T NIL VAL))
                    (RETURN (AND VAL (SELECTQ TYPE
                                              (INPUT (APPEND (SETQ TEM (LISPXGETINPUT
                                                                        (COND
                                                                           ((NULL (CAR VAL))
                                                                            (LISTGET1 VAL '*HISTORY*))
                                                                           (T (CAR VAL)))
                                                                        VAL))
                                                             (AND (NEQ (CAR (LAST TEM))
                                                                       HISTSTR0)
                                                                  (LIST HISTSTR0))))
                                              (ENTRY VAL)
                                              (ENTRIES (LIST VAL))
                                              (COPY (LISPXFIND1 VAL))
                                              (COPIES (LIST (LISPXFIND1 VAL)))
                                              (GO BAD))))
                BAD (ERROR TYPE '"- LISPXFIND ?" T))))))
```

(LISPXGETINPUT

```
  (LAMBDA (INPUT EVENT)                                    (* |separate| |function| |so| |can| |be| |advised|)
    INPUT))
```

(REMEMBER

```
  (LAMBDA (LINE)                                           (* |wt:| "28-FEB-79 23:52")
    (MARKASCHANGED (GETEXPRESSIONFROMEVENTSPEC LINE)
           'EXPRESSIONS)))
```

(GETEXPRESSIONFROMEVENTSPEC

```
(LAMBDA (LINE)                                                    (* |wt:| "28-FEB-79 23:49")
  (PROG ((INPUTLINES (LISPXFIND LISPXHISTORY LINE 'INPUT T))
         NEXT LL)
        (SETQ LL (|while| (SETQ NEXT (FMEMB HISTSTR0 INPUTLINES))
                    |collect| (SETQ LL (LDIFF INPUTLINES NEXT))
                              (SETQ INPUTLINES (CDR NEXT))
                              (COND
                                 ((EQ (CAR LL)
                                      'RETRY\:)
                                  (SETQ LL (CDR LL))))
                              (SETQ LL (COND
                                          ((NULL (CDR LL))
                                           (CAR LL))
                                          (T (SELECTQ (ARGTYPE (CAR LL))
                                                  ((1 3)
                                                       (CONS (CAR LL)
                                                             (COND
                                                                ((CDDR LL)
                                                                 (CDR LL))
                                                                (T (CADR LL)))))
                                                  (COND
                                                     ((CDDR LL)
                                                      (ERROR LL "Can't remember"))
                                                     ((EQ (CAR LL)
                                                          'SET)
                                                      (CONS 'SETQ (CONS (CAADR LL)
                                                                        (MAPCAR (CDADR LL)
                                                                                (FUNCTION KWOTE)))))
                                                     (T (CONS (CAR LL)
                                                              (MAPCAR (CADR LL)
                                                                      (FUNCTION KWOTE))))))))))
                              LL))
        (RETURN (MKPROGN LL)))))
```

(LISPXFIND0

```
(LAMBDA (LINE0 LST INDEX MOD)                                     (* |lmm| "10-MAY-81 20:57")
```

(* |Value| |is| \a |list| |of| |entries| |on| |history| |list.| |lispxfind| |decides| |whatto| |do| |with| |them.|)

```
  (PROG (HISTORYFLG THRUFLG L1 L2 TEM)
        (COND
           ((NULL (CDR LINE0))
            (GO OUT)))
        (SELECTQ (CAR LINE0)
             (@
```

(* E.\g. REDO @ FOO\, |same| |as| |retrieve| FOO |and| |then| REDO |it,| |except| |don't| |get| |two| |copies| |of| FOO |on| HISTORY |list.|)

```
                (COND
                   ((NULL (SETQ LINE0 (GETPROP (SETQ TEM (CADR LINE0))
                                               '*HISTORY*)))
                    (ERROR TEM '" ?" T)))
                (RETURN (COND
                           ((EQ TYPE 'INPUT)
```

(* CADR |is| |the| |input,| CDDR |the| |events| |themselves.| |Note| |that| |input| |may| |correspond| |to| |the| |history| |portion,| |e.g.| |user| |says| NAME FOO USE. |The| LIST LIST |is| |because| |value| |of| LSPXFIND0 |is| |supposed| |to| |be| \a |list| |of| |events.|)

```
                            (LIST (LIST (CADR LINE0))))
                           (T (CDDR LINE0)))))
             ((FROM |from|)
```

(* |Input| |can| |be| |of| |form| -
FROM |...| TO |...| |or| |...| TO |...| -
FROM |...| THRU |...| |or| |...| THRU |...| -
FROM ...\; TO ...\; THRU ...\; |or| \a |list| |of| |entries.|)

```
                 (SETQ L1 (CDR LINE0)))
             ((TO THRU |to| |thru|)
                 (SETQ THRUFLG (OR (EQ (CAR LINE0)
                                       'THRU)
                                   (EQ (CAR LINE0)
                                       '|thru|)))
                 (SETQ L2 (HISTORYFIND LST INDEX MOD (CDR LINE0)
                                  LINE))
                 (GO LDIFF))
             ((ALL |all|)
                 (RETURN (HISTORYFIND LST INDEX MOD LINE0 LINE)))
             NIL)                                                 (* A\t |this| |point| |we| |know| |it| |did| |not| |begin| |with| TO |or| THRU.)
        (COND
           ((AND (OR (SETQ TEM (FMEMB 'TO LINE0))
```

```
                    (SETQ TEM (FMEMB '|to| LINE0))
                    (SETQ THRUFLG (SETQ TEM (OR (FMEMB 'THRU LINE0)
                                                (FMEMB '|thru| LINE0)))))
                (NEQ (CAR (NLEFT LINE0 1 TEM))
                     'F))
           (SETQ L1 (HISTORYFIND LST INDEX MOD (LDIFF (OR L1 LINE0)
                                                      TEM)
                                 LINE))
           (SETQ L2 (HISTORYFIND LST INDEX MOD (CDR TEM)
                                 LINE)))
          (L1                                                    (* |Line| |began| |with| FROM\, |but| |did| |not| |contain| \a TO |or|
              THRU.)
              (SETQ L1 (HISTORYFIND LST INDEX MOD L1 LINE)))
          (T (GO OUT)))
    LDIFF
        (RETURN (COND
                   ((NULL L1)
                    (AND THRUFLG (SETQ L2 (CDR L2)))
                    (LDIFF LST L2))
                   ((NULL L2)
                    (DREVERSE (COND
                                 ((NULL (CDR L1))
                                  (APPEND LST))
                                 (T (LDIFF LST (CDR L1))))))
                   ((TAILP L2 L1)
                    (AND THRUFLG (SETQ L2 (CDR L2)))
                    (LDIFF L1 L2))
                   (T (AND (NULL THRUFLG)
                           (SETQ L2 (CDR L2)))
                      (DREVERSE (COND
                                   ((NULL (CDR L1))
                                    (APPEND L2))
                                   (T (LDIFF L2 (CDR L1)))))))))
    OUT (SETQ TEM (CAR (HISTORYFIND LST INDEX MOD LINE0 LINE)))
        (RETURN (LIST (COND
                         ((AND HISTORYFLG (EQ TYPE 'INPUT))
                          (CONS (LISTGET1 TEM '*HISTORY*)
                                (CDR TEM)))
                         (T TEM)))))))
```

(LISPXFIND1

```
(LAMBDA (X)
```

(* |Produces| \a |copy| |of| \a |history| |entry| |so| |that| |if| |the| |history| |list| |recycles,| |and| |this| |entry| |is| |cannibalized,| |the| |value| |of| LISPXFIND1 |is| |not| |touched.|)

```
  (CONS (APPEND (CAR X))
        (CONS (CAR (SETQ X (CDR X)))
              (CONS (CAR (SETQ X (CDR X)))
                    (CDR X))))))
```

(HISTORYFIND

```
(LAMBDA (LST INDEX MOD EVENTADDRESS LISPXFINDFLG)              (* |wt:| " 9-SEP-78 23:25")
```

(* |Searches| \a |history| |list| |and| |returns| |the| |tail| |for| |which| |car| |is| |the| |indicated| |entry.|)

```
  (PROG ((L LST)
         (X0 EVENTADDRESS)
         Z TEM _FLG
         =FLG VAL PREDFLG ALLFLG)
    LP   (SELECTQ (SETQ Z (CAR EVENTADDRESS))
                  (\\ (SETQ L (AND (EQUAL (CAAAR LASTHISTORY)
                                          (CDR LASTHISTORY))
                                   (CAR LASTHISTORY))))
                  ((ALL |all|)
                      (COND
                         ((NULL LISPXFINDFLG)                  (* ALL |only| |interpreted| |on| |calls| |from| |lispxfind.|)
                          (ERROR Z '" ?" T)))
                      (SETQ ALLFLG T)
                      (SETQ EVENTADDRESS (CDR EVENTADDRESS))
                      (GO LP))
                  (= (SETQ =FLG T)
                     (SETQ EVENTADDRESS (CDR EVENTADDRESS))
                     (GO LP))
                  (_ (SETQ _FLG
                           T)
                     (SETQ EVENTADDRESS (CDR EVENTADDRESS))
                     (GO LP))
                  ((F \f)
                      (COND
                         ((SETQ TEM (CDR EVENTADDRESS))
```

(* |Otherwise,| F |is| |not| \a |special| |symbol,| |e.g.| |user| |types| REDO F\, |meaning| |search| |for| F |itself.|)

```
                    (SETQ EVENTADDRESS (CDR EVENTADDRESS))
```

```
                    (SETQ Z (CAR EVENTADDRESS))))
                (HISTORYFIND1))
            ((SUCHTHAT |suchthat|)
```

(* |What| |follows| SUCHTHAT |is| \a |functionto| |be| |applied| |to| |two| |arguments,| |input| |portion,| |and| |entire| |event,| |and| |if| |true,| |approves| |that| |event.| |can| |be| |used| |in| |conjuncton| |with| ALL |or| _.)

```
             (SETQ PREDFLG T)
             (SETQ EVENTADDRESS (CDR EVENTADDRESS))
             (SETQ Z (CAR EVENTADDRESS))
             (HISTORYFIND1))
         (COND
            ((OR _FLG
                 =FLG
                 (NOT (NUMBERP Z)))
             (HISTORYFIND1)                                    (* |Does| |searching.|)
             )
            ((ILESSP Z 0)
```

(* |Entries| |on| LST |are| |numbered| |starting| |at| INDEX |and| |decreasing| |by| 1 |if| Z |is| |negative,| |count| |back| |corresponding| |number.| |if| Z |is| |positive,| |count| |forward,| |except| |when| Z |is| |first| |member| |on| X |in| |which| |case| Z |is| |the| |absolute| |event| |address,| |i.e.| Z |refers| |to| |the| |index| |that| |would| |be| |printed| |by| |the| ?? |command.|)

```
             (SETQ L (NTH L (IMINUS Z))))
            ((NEQ L LST)                                       (* |move| |forward.|)
             (SETQ L (NLEFT LST (ADD1 Z)
                            L)))
            ((NOT (IGREATERP Z INDEX))
             (SETQ L (CDR (NTH L (IDIFFERENCE INDEX Z)))))
            ((IGREATERP (SETQ TEM (IPLUS INDEX MOD (IMINUS Z)))
                    0)
```

(* E.\g. |Suppose| |history| |numbers| |have| |just| |'RECYCLED',| |i.e.| |current| |history| |is| 5\, |and| |user| |references| 97 |must| |subtract| 97 |from| 105 |to| |find| |how| |far| |back| |the| |entry| |is.| |The| IGREATERP |check| |is| |in| |case| |user| |simply| |typed| |very| |large| |number.|)

```
              (SETQ L (CDR (NTH L TEM))))))
        (COND
           ((NULL L)
            (COND
               (ALLFLG (RETURN VAL))
               ((AND DWIMFLG LISPXFINDFLG (SOME LINE (FUNCTION (LAMBDA (EVENTADDRESS TAIL)
                                                                 (AND (NOT (FMEMB EVENTADDRESS LISPXFINDSPLST)
                                                                           )
                                                                      (FIXSPELL EVENTADDRESS 70 LISPXFINDSPLST
                                                                             T TAIL))))))
```

(* O\n |calls| |from| LISPXFIND\, |attempt| |to| |find| \a |misspelling| |in| |the| |line,| |and| |if| |so,| |do| \a |retfrom.|)

```
                (RETFROM 'LISPXFIND (LISPXFIND HISTORY LINE TYPE BACKUP QUIETFLG))))
            (ERROR Z '" ?" T))
           ((NULL (SETQ EVENTADDRESS (CDR EVENTADDRESS)))
            (SETQ LASTHISTORY (CONS L (CONS (CAR (SETQ TEM (CAAR L)))
                                            (CDR TEM))))
```

(* |For| \\ |command.| |Input| |is| |copied| |so| |that| |it| |can| |be| |used| |as| \a |check| |to| |see| |whether| |this| |particular| |event| |has| |been| |recycled| |since| |it| |was| |last| |referenced.|)

```
            (COND
               ((NULL ALLFLG)
                (RETURN L))
               (T (SETQ VAL (NCONC1 VAL (CAR L)))
                  (SETQ EVENTADDRESS X0)))))
        (SETQ L (CDR L))
        (SETQ _FLG
         NIL)
        (SETQ =FLG NIL)
        (SETQ PREDFLG NIL)
        (SETQ HISTORYFLG NIL)
        (GO LP))))
```

(HISTORYFIND1

```
  (LAMBDA NIL                                                  (* |rmk:| "27-MAY-82 23:11")
```

(* |SEarches| |history| |list,| |forward| |or| |backward,| |depending| |on| _FLG\, |looking| |for| Z (|bound| |in| |historyfind|)\, |and| |resetting| L |to| |the| |corresponding| |tail.|)

```
    (PROG (PAT1 PAT2 TEM PRED)
          (AND _FLG
               (COND
                  ((EQ L LST)
                   (SETQ L (LAST L)))
                  (T (SETQ L (NLEFT LST 2 L)))))
          (COND
             (PREDFLG)
             ((AND (ATOM Z)
```

```
                  (EQ (CHCON1 Z)
                      (CHARCODE _)))
              (SETQ PAT1 (EDITFPAT (PACK (CDR (DUNPACK Z CHCONLST1)))
                                   T)))
           (T (SETQ PAT2 (EDITFPAT Z T))))
     LP  (COND
           ((COND
               ((AND (OR (EQ (SETQ TEM (CAAAR L))
                             'UNDO)
                         (EQ TEM '|undo|))
                     (EQ (CADDAR L)
                         (QUOTE)))                          (* UNDO |events| |that| |failed| |to| |find| |are| |ignored.|)
                NIL)
               ((AND (SETQ TEM (LISTGET1 (CAR L)
                                         '*HISTORY*))
                     PAT2
                     (OR (EQ PAT2 (CAR TEM))
                         (EQ PAT2 (CAR (LISTP (CAR TEM))))))
                (SETQ HISTORYFLG T))
               (PREDFLG (APPLY* Z (CAAR L)
                                  (CAR L)))
               (PAT1 (EDIT4E PAT1 (CAAAR L)))
               (T (HISTORYMATCH (COND
                                  (=FLG (COND
                                          ((FMEMB '*HISTORY* (CAR L))
                                                            (* |The| |value| |slot| |is| |bell| -
                                                            |and| |is| |meaningless.|)
                                           (SETQ L (CDR L))
                                           (GO LP1))
                                          ((AND (FMEMB '*PRINT* (CAR L))
                                                (OR (EQ (SETQ TEM (CAAAR L))
                                                        'OK)
                                                    (EQ TEM 'EVAL)))

           (* |Although| |the| |value| |of| |this| |event| |may| |match| |the| |pattern,| |the| |user| |never| |saw| |the| |value| |printed| |out|
           (|and| PRINTHISTORY) |wouldnt| |print| |it| |out.|)

                                                           (SETQ L (CDR L))
                                                           (GO LP1)))
                                                 (CADDAR L))
                                          ((AND (NULL REREADFLG)
                                                (NULL (CAAR L)))
                                           (LISTGET1 (CAR L)
                                                     '*HISTORY*))
                                          (T (CAAR L)))
                                  PAT2
                                  (CAR L))))
               (RETURN L))
              (_FLG
               (SETQ L (NLEFT LST 1 L)))
              (T (SETQ L (CDR L))))
     LP1 (COND
           ((NULL L)
            (RETURN NIL)))
         (GO LP))))
```

(HISTORYMATCH

```
(LAMBDA (INPUT PAT EVENT)
  (EDITFINDP INPUT PAT T)))
```

(VALUEOF

```
(NLAMBDA LINE                                               (* |wt:| "29-OCT-78 22:25")
```

(* |the| |problem| |is| |how| |to| |decide| |whether| |or| |not| |the| |last| |event| |is| |to| |be| |considered| |in| |interpreting| |the|
|history| |specificaton.| |if| |the| |use| |typed,| (VALUEOF -1)\, |he| |obviously| |doesnt| |want| |this| |event| |considered.|
|on| |the| |other| |hand,| |if| |user| |types| |to| |editor| (I \: (VALUEOF -1)) |he| |does| |want| |mos| |recent| |event| |considered.|
VALUEOF |simply| |uses| |the| |appearance| |of| VALUEOF |in| |the| |event| |as| |an| |indicator.|
|however,| \a |separate| |function| VALUOF |is| |provided| |so| |that| |users,| |e.g.|
|kaplan,| |can| |define| |lispxmacros| |which| |effectively| |call| VALUEOF.)

```
  (VALUOF LINE (EDITFINDP (CAAAR LISPXHISTORY)
                          'VALUEOF))))
```

(VALUOF

```
(LAMBDA (LINE BACKUP)                                       (* |lmm| " 1-May-86 23:20")
  (DECLARE (SPECVARS LINE BACKUP HISTORYFLG))
  (PROG (Y HISTORYFLG)
        (SETQ Y (COND
                   ((NULL LINE)
                    (CADAR LISPXHISTORY))
                   (T (CAR (HISTORYFIND (COND
                                          (BACKUP (SETQ Y (SUB1 (CADR LISPXHISTORY)))
                                                  (CDAR LISPXHISTORY))
                                          (T (SETQ Y (CADR LISPXHISTORY))
```

```
                                        (CAR LISPXHISTORY)))
                                  Y
                                  (OR (CADDDR LISPXHISTORY)
                                      100)
                                  (MKLIST LINE))))))
        (RETURN (VALUOF-EVENT Y)))))
```

(VALUOF-EVENT

```
(LAMBDA (Y)                                                    (* |lmm| " 1-May-86 23:20")
  (COND
     ((NULL (SETQ LINE (LISTGET1 Y '*GROUP*)))
      (CL:VALUES-LIST (LISTGET (CDDDR Y)
                               'LISPXVALUES)))
     ((NULL (CDR LINE))
      (VALUOF-EVENT (CAR LINE)))
     (T (|for| X |in| LINE |collect| (VALUOF-EVENT X))))))
```

(LISPXUSE

```
(LAMBDA (LINE HISTORY LSPXHST)                                 (* |wt:| 18-AUG-76 10 31)
  (PROG (EXPR ARGS VARS STATE LST TEM USE-ARGS GENLST LISPXHIST)
                                                               (* LISPXHIST |rebound| |to| NIL |so| ESUBST |doesn't| |put|
                                                               |any| |side| |information| |on| |history.|)
        (COND
           ((NULL LINE)
            (ERROR '"use what??" '\` T)))
        (SETQ STATE 'VARS)
    LP                                                         (* |Parses| |input| |string| |using| \a |finite| STATE |machine.|)
        (COND
           ((OR (NULL LST)
                (NULL (CDR LINE))
                (NULL (SELECTQ (CAR LINE)
                               ((FOR |for|)
                                    (COND
                                       ((EQ STATE 'VARS)
                                        (SETQ VARS (NCONC1 VARS LST))
                                        (SETQ TEM (APPEND LST TEM))
                                        (SETQ STATE 'ARGS)
                                        (SETQ LST NIL)
                                        T)))
                               ((AND |and|)
                                    (COND
                                       ((EQ STATE 'EXPR)
                                        NIL)
                                       (T (COND
                                             ((EQ STATE 'ARGS)
                                              (SETQ ARGS (NCONC1 ARGS LST)))
                                             ((EQ STATE 'VARS)     (* E.\g. |user| |types| USE A AND B |following| |previous| USE
                                                                   |command.|)
                                              (SETQ VARS (NCONC1 VARS LST))))
                                          (SETQ STATE 'VARS)
                                          (SETQ LST NIL)
                                          T)))
                               ((IN |in|)
                                    (COND
                                       ((AND (EQ STATE 'VARS)
                                             (NULL ARGS))
                                        (SETQ VARS (NCONC1 VARS LST))
                                        (SETQ TEM (APPEND LST TEM))
                                        (SETQ STATE 'EXPR)
                                        (SETQ LST NIL)
                                        T)
                                       ((EQ STATE 'ARGS)
                                        (SETQ ARGS (NCONC1 ARGS LST))
                                        (SETQ STATE 'EXPR)
                                        (SETQ LST NIL)
                                        T)))
                               NIL)))
            (SETQ LST (NCONC1 LST (CAR LINE)))
            (COND
               ((MEMBER (CAR LINE)
                        TEM)
                (SETQ GENLST (CONS (CONS (CAR LINE)
                                         (GENSYM))
                                   GENLST))
```

(* |This| |enables| USE A B FOR B A\, USE A FOR B AND B FOR A\, |or| USE A FOR B AND B C FOR A)

```
               ))))
        (COND
           ((SETQ LINE (CDR LINE))
            (GO LP)))
        (SELECTQ STATE
            (VARS (SETQ VARS (NCONC1 VARS LST)))
            (ARGS (SETQ ARGS (NCONC1 ARGS LST)))
            (EXPR (SETQ EXPR LST))
```

```
     (HELP))                                                    (* ARGS |and| VARS |are| |lists| |of| |lists.|)
(AND (NULL EXPR)
     ARGS
     (SETQ EXPR (LIST 'F (CAAR ARGS))))

(* EXPR |specifies| |expressions| |to| |be| |substituted| INTO. E.\g.
USE FOO FOR FIE IN FUM |or| USE FOO FOR FIE. I\n |latter| |case,| |searches| |for| FIE.
|The| F |is| |added| |because| |of| |numbers,| |e.g.| USE 3 FOR 4 |means| |find| 4\, |whereas| USE FOO FOR FIE IN 4
|means| |the| 4TH |expression| |back.|)

(AND (NULL ARGS)
     (SETQ USE-ARGS (CADR (FMEMB 'USE-ARGS (LISPXFIND HISTORY EXPR 'ENTRY T T)))))
(SETQ EXPR (LISPXFIND HISTORY EXPR 'INPUT T T))             (* EXPR |now| |is| |the| |expression|
                                                           (\s) |to| |be| |substituted| |into.|)
(COND
   (ARGS                                                   (* |Arguments| |specifically| |named| |by| |user,| |i.e.|
                                                           USE |...| FOR |...|)
         (SETQ USE-ARGS (CONS ARGS EXPR))                  (* T\o |be| |saved| |in| |case| |user| |gives| \a |use| |command|
                                                           |referring| |to| |this| |event.|)
         (SETQ EXPR (LIST EXPR)))
   (USE-ARGS                                               (* |Arguments| |specified| |by| |other| USE |command.|)
          (SETQ ARGS (CAR USE-ARGS))
          (SETQ EXPR (LIST (CDR USE-ARGS)))
          (COND
             ((AND (CDR ARGS)
                   (NULL (CDR VARS)))

(* |User| |types| |command| |of| |the| |form| USE A FOR B AND C FOR D |and| |follows| |this| |with| USE E F.)

              (SETQ VARS (MAPCAR (CAR VARS)
                                 (FUNCTION CONS))))))
   ((OR (CDR VARS)
        (CDR (FMEMB HISTSTR0 EXPR)))

(* |More| |than| |one| |operation,| |but| |no| ARGS. |e.g.| USE FOO IN A AND B\, |or| |else| |multiple| |arguments| |specified|
|in| |the| |referent| |operation,| |e.g.| |it| |was| |of| |the| |form| USE A FOR B AND C FOR D.)

    (ERROR '"for what ?" '\` T))
   (T                                                      (* E.\g. LOAD (FOO) |followed| |by| USE MAKEFILE
                                                           RECOMPILE.)
      (SETQ TEM (COND
                   ((CDDR EXPR)
                    (CAR EXPR))
                   (T (CAAR EXPR))))
      (SETQ ARGS (LIST (LIST TEM)))
      (SETQ EXPR (LIST EXPR))))
(SETQ TEM (LISPXUSE0 VARS ARGS EXPR GENLST))
(NCONC LSPXHST (LIST 'USE-ARGS USE-ARGS))
(RETURN TEM))))
```

(LISPXUSE0

```
(LAMBDA (VARS ARGS EXPR GENLST)                             (* |wt:| 24-JUN-76 14 19)

(* |Does| |the| |actual| |substitution| |after| LISPXUSE |has| |computed| |the| VARS\, ARGS\, |and| EXPRS.
VARS |is| \a |list| |of| |lists| |of| |variables,| |the| |extra| |list| |corresponding| |to| |the| |clauses| |of| |an| AND\, |e.g.|
USE A B FOR C AND D E FOR F |would| |have| ((A B) (D E)) |for| VARS\, |and|
((C) (F)) |for| AAGS.)

  (PROG (VAL)
    LP
(* |Argument| |names| |have| |either| |been| |supplied| |by| |user| |or| |obtained| |implicitly| |from| |another| USE |command.|)

(SETQ EXPR (LISPXUSE1 (CAR VARS)
                      (CAR ARGS)
                      EXPR))
(SETQ VARS (CDR VARS))
(COND
   ((SETQ ARGS (CDR ARGS))
    (GO LP))
   (VARS (ERROR '"use what??" '\` T)))
(MAPC GENLST (FUNCTION (LAMBDA (X)
                         (LISPXSUBST (CAR X)
                                     (CDR X)
                                     EXPR T))))
(SETQ VAL (MAPCONC EXPR (FUNCTION (LAMBDA (X)
                                    X))))

(* |Samples:| USE A B C D FOR X Y |means| |substitute| A FOR X AND B FOR Y AND |then| |do| |it| |again| |with| C FOR X
AND D FOR Y. |This| |is| |equivalent| |to| USE A C FOR X AND B D FOR Y |except| |that| |first| |case| |can| |be| |followed|
|by| USE E F |and| |will| |automatically| |substitute| FOR X AND Y.
-
USE A B C FOR D AND X Y Z FOR W |means| |three| |operations,| |with| A FOR D AND X FOR W |in| |the| |first,| B FOR D
AND Y FOR W |in| |the| |second,| |etc.| -
USE A B C FOR D AND X FOR Y |means| |three| |operations,| |first| |with| A FOR D AND X FOR Y |second| |with| B FOR D
AND X FOR Y |etc.| |equivalent| |to| USE X FOR Y AND A B C FOR D.
-
USE A B C FOR D AND X Y FOR Z |causes| |error.| -
```

USE A B FOR B A |will| |work| |correctly,| |but| USE A FOR B AND B FOR A |will| |result| |in| |all| |B's| |being| |changed| |to| |A's.| |The| |general| |rule| |is| |substitution| |proceeds| |from| |left| |to| |right| |with| |each| |'AND'| |handled| |separately.| |Whenever| |the| |number| |of| |variables| |exceeds| |the| |number| |of| |expressions| |available,| |the| |expressions| |multiply.|)

```
        (RETURN VAL))))
```

(LISPXUSE1

```
(LAMBDA (VARS ARGS EXPRS)
  (PROG ((V VARS)
         (A ARGS)
         (E (COPY EXPRS))
         L VFLG AFLG EFLG TEM)
        (SETQ L E)
    LP  (COND
           ((AND GENLST (SETQ TEM (SASSOC (CAR V)
                                          GENLST))
                 (STRPOS 'Ÿ (CAR A)))
            (ERROR '"sorry, that's too hard." (QUOTE)
                   T)))
        (RPLACA E (COND
                     ((EQ (CAR V)
                          '!)
                      (SETQ V (CDR V))
                      (LSUBST (CAR V)
                              (CAR A)
                              (CAR E)))
                     (T (LISPXSUBST (OR (CDR TEM)
                                        (CAR V))
                                    (CAR A)
                                    (CAR E)
                                    T))))
        (COND
           ((NULL (SETQ V (CDR V)))
            (SETQ VFLG T)))
        (COND
           ((NULL (SETQ A (CDR A)))
            (SETQ AFLG T)))
        (COND
           ((AND A V)
            (GO LP))
           ((SETQ E (CDR E))
            (GO LP1)))
        (COND
           ((AND (NULL A)
                 (NULL V))
            (RETURN L)))
        (SETQ EFLG T)
        (SETQ L (NCONC L (SETQ E (COPY EXPRS))))
    LP1 (COND
           ((AND EFLG VFLG AFLG)
            (ERROR '"huh??" (QUOTE)
                   T)))
        (COND
           ((NULL V)
            (SETQ V VARS)))
        (COND
           ((NULL A)
            (SETQ A ARGS)))
        (GO LP))))
```

(LISPXSUBST

```
(LAMBDA (X Y Z CHARFLG)
```

(* |used| |by| |lispx,| |lispxuse| |and| |lispxuse0.| \a |separate| |function| |so| |can| |be| |advised| |for| |applications| |involving| |history| |lists| |contaiing| |different| |types| |of| |inputs,| |e.g.| |strings.|)

```
    (COND
       ((NULL CHARFLG)
        (SUBST X Y Z))
       (T (ESUBST X Y Z T)))))
```

(LISPXUSEC

```
(LAMBDA (LINE HISTORY)                                    (* |lmm| " 7-MAY-82 19:30")
```

(* A |short| |version| |of| |the| USE |command.| $ X Y |is| |equivalent| |to| USE Y FOR X.
|user| |can| |also| |say| $ X = Y |or| $ X -> Y |or| $ Y FOR X. |User| |can| |specify| |event| |with| IN.
|However,| |the| |distributivity| |of| USE |command| |is| |not| |allowed.|
(|Note| |that| $ |can| |be| |ued| |even| |if| |character| |editing| |is| |not| |being| |performed,| |e.g.|
$ FOO FIE |is| |probably| |easier| |to| |type| |than| USE FIE FOR FOO.) I\f |the| |event| |referred| |to| |contains| |an| ERROR |property,| $ |first| |performs| |the| |substitution| |on| |that| |argument,| |and| |then| |substitues| |the| |corrected| |offender| |into| |the| |expression.| I\f |the| |user| |omits| \a |second| |argument,| |e.g.|
|types| $ FOO\, |the| |substitution| |is| |performed| |for| |the| |offender.|
(I\n |this| |case| |there| |must| |be| |an| ERROR |property.|))

```
(PROG (LISPY LISPZ LISPXTEM LISPX1 LISPX2 LISPXIN LISPXHIST) (* LISPXHIST |rebound| |to| NIL |so| ESUBST |doesn't| |put|
                                                              |any| |side| |information| |on| |history.|)
      (COND
         ((CDR (SETQ LISPXIN (FMEMB 'IN LINE)))

      (* |May| |be| |of| |the| |form| $ X IN -- |or| $ X Y IN --. |Note| |that| --
      |may| |specify| \a |group.|)

          (SETQ LINE (LDIFF LINE LISPXIN))
          (SETQ LISPY (LISPXFIND HISTORY (SETQ LISPXIN (CDR LISPXIN))
                                 'ENTRY T T)))
         ((NULL (CDR LINE))                                   (* |Form| |is| |just| $ X.)
          (SETQ LISPY (LISPXFIND HISTORY NIL 'ENTRY T T))))
      (COND
         ((NULL (CDR LINE))
          (COND
             ((SETQ LISPZ (CDR (FMEMB '*ERROR* LISPY)))
              (SETQ LISPX1 (CAR LINE))
              (SETQ LISPX2 (CAR LISPZ))
              (GO OUT))
             ((NUMBERP (CAR LINE))
              (RETURN (LISPXUSEC (CONS 'IN LINE)
                                 HISTORY)))
             (T

      (* |Since| |no| |second| |argument| |was| |specified,| |this| |has| |to| |be| |an| ERROR |correction.|
      |Note| |that| |it| |may| |have| |been| |of| |the| |form| $ X |or| $ X IN --.)

                (PRIN1 '"Unable to figure out what you meant in:" T)
                (PRINTHISTORY1 LISPY T)
                (ERROR!)))))                                  (* |Identify| |substituTEE| |and| |substituTOR.|)
      (COND
         ((CDDR LINE)
          (SELECTQ (CADR LINE)
               ((TO = ->)
                   (SETQ LISPX1 (CADDR LINE))
                   (SETQ LISPX2 (CAR LINE)))
               (FOR (SETQ LISPX1 (CAR LINE))
                    (SETQ LISPX2 (CADDR LINE)))
               (ERROR (CADR LINE)
                      '" ?" T)))
         (T (SETQ LISPX1 (CADR LINE))
            (SETQ LISPX2 (CAR LINE))))
      (COND
         ((NULL LISPY)                                        (* |Form| |of| |command| |is| $ X Y.
                                                              |Search| |for| X.)
          (SETQ LISPXTEM (COND
                             ((AND (NLISTP LISPX1)
                                   (NLISTP LISPX2)
                                   (NOT (STRPOS 'Ÿ LISPX2)))
                              (PACK (LIST 'Ÿ LISPX2 'Ÿ)))
                             (T LISPX2)))
          (SETQ LISPY (LISPXFIND HISTORY (SETQ LISPXIN (LIST LISPXTEM))
                                 'ENTRY T T))))
      (SETQ LISPZ (CDR (FMEMB '*ERROR* LISPY)))

      (* T\o |see| |if| |the| |event| |contains| |an| |error| |property.| |Note| |that| |even| |if| |the| |user| |identifies| |an| |event| |using|
      IN\, |if| |the| |event| |contains| |an| |offender,| |the| |character| |substitution| |takes| |place| |only| |in| |the| |error,| |not| |in| |the|
      |whole| |expression.| |See| |comment| |below| |after| EDITFINDP.)

  OUT (SETQ LISPY (COPY (LISPXFIND HISTORY LISPXIN 'INPUT T T)))

      (* |Need| |another| |call| |to| LISPXFIND |even| |though| |we| |already| |have| |the| |entry| |because| LISPXFIND |contains|
      |smarts| |about| |what| |fields| |to| |extract,| |e.g.| |did| |use| |say| $ X Y IN USE |or| $ X Y IN -1\, |etc.|)

      (COND
         ((NULL LISPZ)

      (* |The| |user| |is| |using| $ |to| |avoid| |having| |to| |type| |alt-modes| |around| |his| |patterns,| |otherwise| |this| |is| |essentially|
      \a |simplified| USE |command.| |therefore| |perform| |the| |substitution| |in| |the| |input,| |i.e.|
      LISPY)

          (GO OUT1)))                                         (* |There| |was| |an| |error| |in| |the| |indicated| |event.|)
      (SETQ LISPZ (CAR LISPZ))
      (COND
         ((AND (EQ LISPX2 LISPZ)
               (NUMBERP LISPX1))

      (* $ 1 2 |will| |change| |all| |1's| |to| |2's| |occurring| |inside| |of| |other| |atoms| |or| |strings.|
      I\t |will| |not| |change| |the| |number| 1 |to| |the| |number| 2.0 |Therefore,| |this| |check| |is| |for| |the| |case| |where| |the| |'bad|
      |guy'| |was| \a |number,| |and| |the| |user| |was| |typing| |in| |the| |correct| |number| |in| |the| |form| |of| $ |number.|
      |this| |frequently| |happens| |for| |correction| |to| EDIT |commands,| |e.g.|
      |user| |types| (|ri| 1 33) |meaning| (|ri| 1 3) |and| |then| |corrects| |by| $ 3)

          (SETQ LISPXTEM LISPX1)
          (AND (NULL EDITQUIETFLG)
               (PRIN2 LISPX2 T T)
```

```
                (PRIN1 '-> T)
                (PRINT LISPX1 T T))

        (* |Since| |in| |all| |other| |cases,| ESUBST |will| |cause| \a |message| |of| |this| |form| |to| |be| |printed,| |we| |also| |do| |it|
        |here| |to| |be| |consistent.|)

            )
           ((NULL LINE)
            (COND
               ((OR (LITATOM LISPZ)
                    (STRINGP LISPZ))

        (* |The| |effect| |of| |this| |is| |to| |cause| |the| |operation| |that| |caused| |the| |error| |to| |be| |reexecuted| |this| |time|
        |searching| |for| |something| |thatis| |'close'| |to| |the| |word| |producing| |the| |error,| |e.g.|
        |user| |types| INSERT -- AFTER CONDD |and| |system| |types| CONDD ? |user| |then| |types| $ |causing| |the| |command|
        INSERT -- AFTER CONDD$$ |to| |be| |executed.|)

                 (SETQ LISPXTEM (PACK (LIST LISPZ 'ŸŸ))))
                (T (ERROR '" ? " (QUOTE)
                          T))))
           ((NULL (NLSETQ (SETQ LISPXTEM (ESUBST LISPX1 LISPX2 (COND
                                                                  ((LISTP LISPZ)
                                                                   (COPY LISPZ))
                                                                  (T LISPZ))
                                                NIL T))))

        (* |The| |indicated| |characters| |do| |not| |appear| |in| LISPZ\, |the| |offender.|
        |Therefore,| |perform| |the| |substitution| |in| |the| |input.|)

            (GO OUT1)))
        (COND
           ((EDITFINDP LISPY LISPZ)
            (RETURN (SUBST LISPXTEM LISPZ LISPY)))
           (T (PRIN2 LISPZ T T)
              (PRIN1 '" does not appear in " T)
              (PRINTHISTORY1 (LIST LISPY)
                     T)
              (ERROR!)))
     OUT1
        (RETURN (ESUBST LISPX1 LISPX2 LISPY T T)))))
```

(LISPXFIX

```
  (LAMBDA (INPUT COMS)                                          (* |wt:| 14-JUL-76 14 38)
    (PROG (LISPXHIST)
          (RETURN (CAR (LAST (EDITL (COND
                                       ((AND (EQ (CADR INPUT)
                                                 HISTSTR0)
                                             (NULL (CDDR INPUT)))
                                                                  (* |eval| |input,mkae| |the| |current| |expression| |be| |the| |form|
                                                                  |itself.|)
                                        (LIST (CAR INPUT)
                                              INPUT))
                                       (T (LIST INPUT)))
                                    COMS)))))))
```

(CHANGESLICE

```
  (LAMBDA (N HISTORY L)                                         (* |wt:| "22-NOV-78 23:27")

        (* |Undoing| \a CHANGSLICE |involves| |another| |call| |to| CHANGESLICE\, |because| |you| |can't| |just| |replace| |the|
        |pointers| |because| |of| |the| |ring| |buffer| |aspect| |of| |the| |history| |list.|
        I\n |other| |words,| |the| |place| |where| |events| |was| |deleted| |may| |now| |be| |the| |beginning| |of| |the| |history| |list.|
        |Therefore,| L |represents| |the| |forgotten| |events| |if| |any,| |in| |the| |case| |that| |the| |history| |list| |is| |being| |enlarged| |by|
        |virtue| |of| |undoing| \a CHANGESLICE.)

    (COND
       ((ILESSP N 3)
        (ERROR N '"is too small"))
       ((NULL HISTORY)
        (AND LISPXHISTORY (CHANGESLICE N LISPXHISTORY))
        (AND EDITHISTORY (CHANGESLICE N EDITHISTORY)))
       (T (NCONC (CAR HISTORY)
                 L)                                             (* |Add| |forgotten| |events,| |if| |any.|)
          (UNDOSAVE (LIST 'CHANGESLICE (CADDR HISTORY)
                          HISTORY
                          (CDR (SETQ L (NTH (CAR HISTORY)
                                            N))))
                 LISPXHIST)
          (FRPLACA (CDDR HISTORY)
                 N)
          (FRPLACA (CDDDR HISTORY)
                 (ITIMES (ADD1 (IQUOTIENT (SUB1 N)
                                          100))
                         100))
          (COND
             (L                                                 (* |Chop| |off| |the| |extra| |events.|)
```

```
                (FRPLACD L)))))
    N))
```

(LISPXSTATE

```
(LAMBDA (NAME STATE)                                    (* STATE |is| |either| |'BEFORE'| |or| |'AFTER'|)
  (PROG (X Y)
        (COND
           ((NULL (SETQ X (GETP NAME 'STATE)))          (* |First| |time| STATE |command| |used| |with| NAME.)
            (COND
               ((NULL (SETQ Y (CDR (GETP NAME '*HISTORY*))))   (* |The| CDR |is| |because| CAR |corresponds| |to| |he|
                                                                |'arguments'|)
                (ERROR NAME '" ?" T))
               ((EQ STATE 'AFTER)
                (RETURN 'WAS)))
            (MAPC Y (FUNCTION UNDOLISPX2))
            (/PUT NAME 'STATE (CONS 'BEFORE LISPXHIST)))
           ((EQ STATE (CAR X))
            (RETURN 'WAS))
           (T (UNDOLISPX2 X)
              (/PUT NAME 'STATE (CONS STATE LISPXHIST))))
        (RETURN STATE))))
```

(LISPXTYPEAHEAD

```
(LAMBDA NIL                                             (* |wt:| 1-JUL-76 14 26)
  (PROG (X L)
    LP  (PRIN1 '> T)
        (NLSETQ (SELECTQ (SETQ X (LISPXREAD T T))
                         ((ŸOK ŸGO)
                             (MAPC L (FUNCTION LISPXUNREAD))
                             (RETFROM 'LISPXTYPEAHEAD))
                         (ŸSTOP (RETFROM 'LISPXTYPEAHEAD))
                         (FIX (SETQ L (EDITE L)))
                         (ŸQ (PRIN1 '\\\\ T)
                            (PRINT (COND
                                       ((NLISTP (SETQ X (CAR L)))
                                        X)
                                       (T (CAR X)))
                                   T T)
                            (SETQ L (CDR L)))
                         (?? (MAPC (REVERSE L)
                                   (FUNCTION (LAMBDA (X)
                                                     (PRINTHISTORY1 (LIST X '>)
                                                            T T)))))
                         (SETQ L (CONS (COND
                                           ((OR (LISTP X)
                                                (NULL (READP T)))
                                            (LIST X))
                                           (T
        (* |The| |extra| |argument| |to| READLINE |is| |so| |that| \a |line| |consisting| |of| |just| ]\, |e.g.|
        FOO] |will| |read| |is| |as| (NIL) |instead| |of| NIL.)

                                                (CONS X (READLINE T NIL T))))
                                       L))))
        (GO LP))))

)

(ADDTOVAR SYSTEMINITVARS (LISPXHISTORY NIL 0 100 100)
                         (GREETHIST))

(DECLARE\: DONTEVAL@LOAD DOCOPY

(RPAQQ \#REDOCNT 3)

(RPAQQ ARCHIVEFLG T)

(RPAQQ ARCHIVEFN NIL)

(RPAQQ ARCHIVELST (NIL 0 50 100))

(RPAQQ DISPLAYTERMFLG NIL)

(RPAQQ EDITHISTORY (NIL 0 30 100))

(RPAQQ HERALDSTRING NIL)

(RPAQQ LASTEXEC NIL)

(RPAQQ LASTHISTORY NIL)

(RPAQQ LISPXBUFS NIL)

(RPAQQ LISPXHIST NIL)
```

```
(RPAQQ LISPXHISTORY (NIL 0 30 100))

(RPAQQ LISPXPRINTFLG T)

(RPAQQ LISPXUSERFN NIL)

(RPAQQ MAKESYSDATE NIL)

(RPAQQ PROMPT#FLG T)

(RPAQQ REDOCNT NIL)

(RPAQQ SYSOUT.EXT SYSOUT)

(RPAQQ SYSOUTFILE WORK)

(RPAQQ SYSOUTGAG NIL)

(RPAQQ TOPLISPXBUFS NIL)
)

(ADDTOVAR LISPXHISTORYMACROS
          (TYPE-AHEAD (LISPXTYPEAHEAD))
          (??T NIL (PROG (TEM)
                         (RESETVARS ((PRETTYTRANFLG T))
                                (RESETFORM (OUTPUT T)
                                       (PRINTDEF (COND ((NULL (CDAR (SETQ TEM (LISPXFIND LISPXHISTORY LISPXLINE
                                                                                         'ENTRY))))
                                                              (CAAR TEM))
                                                             (T (CAR TEM)))
                                                       NIL T)))
                         (TERPRI T)
                         (RETURN NIL))))

(ADDTOVAR LISPXMACROS
          (SHH NIL (COND ((OR (CDR (LISTP LISPXLINE))
                              (AND (FMEMB (LASTC T)
                                          '(\) ]))
                                   (LITATOM (CAR LISPXLINE))))
                         (APPLY (CAR LISPXLINE)
                                (COND ((AND (LISTP (CADR LISPXLINE))
                                            (NULL (CDDR LISPXLINE)))
                                       (CADR LISPXLINE))
                                      (T (CDR LISPXLINE)))))
                         (T (EVAL (COND (LISPXLINE (CAR LISPXLINE))
                                        (T 'SHH))))))
          (RETRIEVE (PROG ((X (GETP (CAR LISPXLINE)
                                    '*HISTORY*))
                           REREADFLG)
                          (COND ((NULL X)
                                 (ERROR (CAR LISPXLINE)
                                        '" ?" T)))
                          (MAPC (CDDR X)
                                (FUNCTION (LAMBDA (X)
                                                  (HISTORYSAVE LISPXHISTORY X))))
                          (RETURN (CAR LISPXLINE))))
          (BEFORE (LISPXSTATE (CAR LISPXLINE)
                              'BEFORE))
          (AFTER (LISPXSTATE (CAR LISPXLINE)
                             'AFTER))
          (OK (RETFROM (OR (STKPOS 'USEREXEC)
                           'LISPX)
                       T T))
          (REMEMBER\: (PROG1 (LET (FILEPKGFLG)
                                  (EVAL (LISPX/ (CAR LISPXLINE))
                                        LISPXID))
                             (MARKASCHANGED (CAR LISPXLINE)
                                    'EXPRESSIONS)))
          (REMEMBER (REMEMBER LISPXLINE)))

(ADDTOVAR LISPXCOMS SHH RETRIEVE BEFORE AFTER OK REMEMBER\: REMEMBER TYPE-AHEAD ??T)

(ADDTOVAR HISTORYCOMS RETRIEVE TYPE-AHEAD)

(ADDTOVAR LISPXFINDSPLST FROM TO THRU SUCHTHAT ALL AND)

(ADDTOVAR BEFORESYSOUTFORMS
          (SETQ SYSOUTDATE (DATE))
          (PROGN (COND ((NULL FILE)
                        (SETQ FILE SYSOUTFILE))
                       (T (SETQ SYSOUTFILE (PACKFILENAME 'VERSION NIL 'BODY FILE))))
                 (COND ((AND (NULL (FILENAMEFIELD FILE 'EXTENSION))
                             (NULL (FILENAMEFIELD FILE 'VERSION)))
                        (SETQ FILE (PACKFILENAME 'BODY FILE 'EXTENSION SYSOUT.EXT))))))

(ADDTOVAR RESETFORMS
          (SETQ READBUF NIL)
```

```
          (SETQ READBUFSOURCE NIL)
          (SETQ TOPLISPXBUFS (OR (CLBUFS T)
                                 TOPLISPXBUFS))
          (COND ((EQ CLEARSTKLST T)
                 (COND ((EQ NOCLEARSTKLST NIL)
                        (CLEARSTK))
                       (T (* |clear| |all| |stack| |pointers| EXCEPT |those| |on| NOCLEARSTKLST.)
                          (MAPC (CLEARSTK T)
                                (FUNCTION (LAMBDA (X)
                                                  (AND (NOT (FMEMB X NOCLEARSTKLST))
                                                       (RELSTK X))))))))
                (T (MAPC CLEARSTKLST (FUNCTION RELSTK))
                   (SETQ CLEARSTKLST NIL))))

(ADDTOVAR HISTORYSAVEFORMS )

(ADDTOVAR LISPXCOMS Ÿ |...| ?? FIX FORGET NAME ORIGINAL REDO REPEAT RETRY UNDO USE |fix| |forget| |name| |redo|
                    |repeat| |retry| |undo| |use|)

(ADDTOVAR SYSTATS
          (LISPXSTATS LISPX INPUTS)
          (UNDOSAVES UNDO SAVES)
          (UNDOSTATS CHANGES UNDONE)
          NIL
          (EDITCALLS CALLS TO EDITOR)
          (EDITSTATS EDIT COMMANDS)
          (EDITEVALSTATS COMMANDS INVOLVING EVALUATING A LISP EXPRESSION)
          (EDITESTATS USES OF AN E COMMAND TYPED IN DIRECTLY)
          (EDITISTATS USES OF AN I COMMAND TYPED IN DIRECTLY)
          (EDITUNDOSAVES EDIT UNDO SAVES)
          (EDITUNDOSTATS EDIT CHANGES UNDONE)
          NIL
          (P.A.STATS P.A. COMMANDS)
          NIL
          (CLISPIFYSTATS CALLS TO CLISPIFY)
          NIL
          (FIXCALLS CALLS TO DWIM)
          (FIXTIME)
          (ERRORCALLS WERE DUE TO ERRORS)
          (DWIMIFYFIXES WERE FROM DWIMIFYING)
          NIL "OF THOSE DUE TO ERRORS:" (TYPEINFIXES WERE DUE TO ERRORS IN TYPE-IN)
          (PROGFIXES WERE DUE TO ERRORS IN USER PROGRAMS)
          (SUCCFIXES1 OF THESE CALLS WERE SUCCESSFUL)
          NIL "OF THE CALLS DUE TO DWIMIFYING:" (SUCCFIXES2 WERE SUCCESSFUL)
          NIL
          (SPELLSTATS OF ALL DWIM CORRECTIONS WERE SPELLING CORRECTIONS)
          (CLISPSTATS WERE CLISP TRANSFORMATIONS)
          (INFIXSTATS OF THESE WERE INFIX TRANSFORMATIONS)
          (IFSTATS WERE IF/THEN/ELSE STATEMENTS)
          (I.S.STATS WERE ITERATIVE STATEMENTS)
          (MATCHSTATS WERE PATTERN MATCHES)
          (RECORDSTATS WERE RECORD OPERATIONS)
          NIL
          (SPELLSTATS1 OTHER SPELLING CORRECTIONS\, E.G. EDIT COMMANDS)
          NIL
          (RUNONSTATS OF ALL SPELLING CORRECTIONS WERE RUN-ON CORRECTIONS)
          NIL
          (VETOSTATS CORRECTIONS WERE VETOED)
          NIL)

(ADDTOVAR NOCLEARSTKLST )

(APPENDTOVAR AFTERSYSOUTFORMS (COND
                                ((LISTP SYSOUTGAG)
                                 (EVAL SYSOUTGAG))
                                (SYSOUTGAG)
                                ((OR (NULL USERNAME)
                                     (EQ USERNAME (USERNAME NIL T)))
                                 (TERPRI T)
                                 (PRIN1 HERALDSTRING T)
                                 (TERPRI T)
                                 (TERPRI T)
                                 (GREET0)
                                 (TERPRI T))
                                (T (LISPXPRIN1 '"****ATTENTION USER " T)
                                   (LISPXPRIN1 (USERNAME)
                                               T)
                                   (LISPXPRIN1 '":
                                                this sysout is initialized for user " T)
                                   (LISPXPRIN1 USERNAME T)
                                   (LISPXPRIN1 '".
                                                " T)
                                   (LISPXPRIN1 '"To reinitialize, type GREET()
                                                " T)))
       (SETINITIALS))

(MAPC SYSTATS (FUNCTION (LAMBDA (X)
```

```
                          (AND (LISTP X)
                               (EQ (GETTOPVAL (CAR X))
                                   'NOBIND)
                               (SETTOPVAL (CAR X)
                                      NIL)))))

(PUTD 'E)

(DEFINEQ
```

(GREET

```
  (LAMBDA (NAME FLG)                                          (* |lmm| "11-Dec-85 17:58")
    (OR (ERSETQ (PROG (FILE)
                      (TAB 0 0 T)
                      (SETQ USERNAME (COND
                                         ((NULL NAME)
                                          (USERNAME NIL T))
                                         (T (COND
                                                ((GETD 'SETUSERNAME)
                                                 (SETUSERNAME NAME)))
                                            (MKATOM NAME))))
                      (|for| X |in| PREGREETFORMS |do| (EVAL X))
                      (AND (SETQ FILE (GREETFILENAME T))
                           (LOAD FILE 'SYSLOAD))                  (* |System| |greeting|)
                      (AND (SETQ FILE (GREETFILENAME USERNAME))
                           (LOAD FILE T))                         (* |User| |greeting|)
                      (|for| X |in| POSTGREETFORMS |do| (EVAL X))
                      (GREET0)
                      (RETURN T)))
        (PRINTOUT T "error during GREET..." T))))
```

(GREET0

```
  (LAMBDA NIL                                                 ; Edited 19-Apr-2023 18:55 by lmm
                                                              ; Edited 19-Mar-2023 09:58 by lmm
                                                              (* |lmm| "28-DEC-82 08:49")
    (COND
       (GREETDATES (LISPXPRIN1 (CL:MULTIPLE-VALUE-BIND (SECONDS MINUTES HOUR DAY MONTH YEAR)
                                   (CL:GET-DECODED-TIME)
                                 (OR (AND (EVENP (LRSH SECONDS 1))
                                          (CDR (SASSOC (CL:FORMAT NIL "~2D-~A" DAY
                                                                  (CL:NTH MONTH
                                                                          '("JAN" "FEB" "MAR" "APR" "MAY" "JUN"
                                                                            "JUL" "AUG" "SEP" "OCT" "NOV"
                                                                            "DEC")))
                                                       GREETDATES)))
                                     (AND (EVENP SECONDS)
                                          (COND
                                             ((AND FIRSTNAME (ILESSP HOUR 6))
                                              '"You're working late tonight")
                                             ((ILESSP HOUR 12)
                                              '"Good morning")
                                             ((ILESSP HOUR 18)
                                              '"Good afternoon")
                                             (T '"Good evening")))
                                     (AND (EVENP SECONDS 3)
                                          "Hello")
                                     '"Hi"))
                             T)
             (COND
                (FIRSTNAME (LISPXPRIN1 '", " T)
                           (LISPXPRIN1 FIRSTNAME T)))
             (LISPXPRIN1 "." T)
             (LISPXTERPRI T)))))

)

(ADDTOVAR PREGREETFORMS (DREMOVE GREETFORM RESETFORMS)
                        (SETQ CONSOLETIME (SETQ CPUTIME (SETQ EDITIME 0)))
                        (SETQ CONSOLETIME0 (CLOCK 0))
                        (SETQ CPUTIME0 (CLOCK 2)))

(ADDTOVAR POSTGREETFORMS (SETINITIALS)
                         (AND EDITCHARACTERS (APPLY 'SETTERMCHARS EDITCHARACTERS)))

(DECLARE\: DONTEVAL@LOAD DOCOPY

(RPAQQ GREETHIST NIL)

(RPAQQ SYSTEMTYPE NIL)

(RPAQQ GREETFORM (LISPXEVAL '(GREET)
                        '_))

(RPAQQ CUTEFLG NIL)

(RPAQQ GREETDATES ((" 1-JAN" . "Happy new year")
```

```
                    ("12-FEB" . "Happy Lincoln's birthday")
                    ("14-FEB" . "Happy Valentine's day")
                    ("22-FEB" . "Happy Washington's birthday")
                    ("15-MAR" . "Beware the Ides of March")
                    ("17-MAR" . "Happy St. Patrick's day")
                    ("18-MAY" . "It's Victoria Day")
                    (" 1-JUL" . "It's Canada Day")
                    ("31-OCT" . "Trick or Treat")
                    (" 5-NOV" . "<boom> it's Guy Fawkes day")
                    ("25-DEC" . "Merry Christmas")))

(RPAQQ USERNAME NIL)

(RPAQQ HOSTNAME NIL)

(RPAQQ CONSOLETIME 0)

(RPAQQ CONSOLETIME0 0)

(RPAQQ CPUTIME 0)

(RPAQQ CPUTIME0 0)

(RPAQQ EDITIME 0)

(RPAQQ FIRSTNAME NIL)

(ADDTOVAR BEFOREMAKESYSFORMS (SETQ RESETFORMS (CONS GREETFORM RESETFORMS))
                             (SETQ MAKESYSDATE (DATE)))

(ADDTOVAR AFTERMAKESYSFORMS (LISPXEVAL '(GREET)
                                       '_))
)

(DEFINEQ

(LISPXPRINT
  (LAMBDA (X Y Z NODOFLG)
    (AND LISPXPRINTFLG LISPXHIST (LISPXPUT '*LISPXPRINT* (LIST (CONS 'PRINT (COND
                                                                              (Z (LIST X Y Z))
                                                                              (Y (LIST X Y))
                                                                              (X (LIST X)))))
                                       T LISPXHIST))
    (AND (NULL NODOFLG)
         (PRINT X Y Z))))

(LISPXPRIN1
  (LAMBDA (X Y Z NODOFLG)
    (AND LISPXPRINTFLG LISPXHIST (LISPXPUT '*LISPXPRINT* (LIST (COND
                                                                ((AND (EQ Y T)
                                                                      (STRINGP X))
                                                                 (* |The| |string| |itself| |will| |be| |stored.|
                                                                 |This| |saves| 3 |cells.|)
                                                                 X)
                                                                (T (CONS 'PRIN1 (COND
                                                                                  (Z (LIST X Y Z))
                                                                                  (Y (LIST X Y))
                                                                                  (X (LIST X)))))))
                                       T LISPXHIST))
    (AND (NULL NODOFLG)
         (PRIN1 X Y Z))))

(LISPXPRIN2
  (LAMBDA (X Y Z NODOFLG)
    (AND LISPXPRINTFLG LISPXHIST (LISPXPUT '*LISPXPRINT* (LIST (COND
                                                                ((AND (EQ Y T)
                                                                      (NLISTP X)
                                                                      (NOT (STRINGP X)))
                                                                 (* |The| |atm| |will| |be| |stored|)
                                                                 X)
                                                                (T (CONS 'PRIN2 (COND
                                                                                  (Z (LIST X Y Z))
                                                                                  (Y (LIST X Y))
                                                                                  (X (LIST X)))))))
                                       T LISPXHIST))
    (AND (NULL NODOFLG)
         (PRIN2 X Y Z))))

(LISPXPRINTDEF
  (LAMBDA (EXPR FILE LEFT DEF TAIL NODOFLG)                    (* |wt:| 11-MAY-76 19 59)
                                                               (* |so| |uer| |can| |prettyprint| |and| |have| |it| |appear| |on|
                                                               |history| |list|)
    (AND LISPXPRINTFLG LISPXHIST (LISPXPUT '*LISPXPRINT* (LIST (LIST 'LISPXPRINTDEF0 EXPR FILE LEFT DEF TAIL))
                                       T LISPXHIST))
```

```
    (AND (NULL NODOFLG)
         (LISPXPRINTDEF0 EXPR FILE LEFT DEF TAIL))))
```

(LISPXPRINTDEF0

```
(LAMBDA (EXPR FILE LEFT DEF TAIL)                                   (* |wt:| 11-MAY-76 19 59)

        (* |this| |function| |is| |necessar| |to| |implement| |lispxprintdef| |because| |printdef| |itself| |doesnt| |take| \a |file| |argument.|)

    (RESETFORM (OUTPUT FILE)
           (PRINTDEF EXPR LEFT DEF TAIL))))
```

(LISPXSPACES

```
(LAMBDA (X Y Z NODOFLG)
  (AND LISPXPRINTFLG LISPXHIST (LISPXPUT '*LISPXPRINT* (LIST (COND
                                                             ((AND (EQ Y T)
                                                                   (EQ X 1))
                                                              '" ")
                                                             (T (CONS 'SPACES (COND
                                                                                (Y (LIST X Y))
                                                                                (X (LIST X)))))))
                                       T LISPXHIST))
  (AND (NULL NODOFLG)
       (SPACES X Y))))
```

(LISPXTERPRI

```
(LAMBDA (X Y Z NODOFLG)
  (AND LISPXPRINTFLG LISPXHIST (LISPXPUT '*LISPXPRINT* (LIST (COND
                                                             ((EQ X T)
                                                              '"
                                                                "")
                                                             (T (CONS 'TERPRI (COND
                                                                                (X (LIST X)))))))
                                       T LISPXHIST))
  (AND (NULL NODOFLG)
       (TERPRI X))))
```

(LISPXTAB

```
(LAMBDA (X Y Z NODOFLG)
  (AND LISPXPRINTFLG LISPXHIST (LISPXPUT '*LISPXPRINT* (LIST (CONS 'TAB (COND
                                                                          (Z (LIST X Y Z))
                                                                          (Y (LIST X Y))
                                                                          (X (LIST X)))))
                                       T LISPXHIST))
  (AND (NULL NODOFLG)
       (TAB X Y Z))))
```

(USERLISPXPRINT

```
(LAMBDA (X FILE Z NODOFLG)                                          (* |wt:| 14-MAY-76 13 8)

        (* |this| |defnition| |can| |be| |movd'd| |to| |any| |user| |function| |whose| |name| |begins| |with| LISPX |to| |make| |it| |work| |like|
        \a |LISPXprining| |function.| |it| |requires| |that| |the| |file| |argument| |be| |the| |second| |argument,| |and| |that| |the| |function|
        |only| |have| |three| |arguments|)

   ((LAMBDA (POS)                                                   (* |This| |has| |the| |avantage| |of| |working| |both| |compiled|
                                                                    |andinterpreted.|)
      (PROG (FN)
            (SETQ FN (STKNAME POS))
            (RELSTK POS)
            (SETQ FN (COND
                        ((NULL (STRPOS 'LISPX FN NIL NIL T))
                         (HELP FN))
                        (T (MKATOM (SUBSTRING FN 6 -1)))))
            (AND LISPXPRINTFLG LISPXHIST (LISPXPUT '*LISPXPRINT* (LIST (CONS FN (NLIST X FILE Z)))
                                               T LISPXHIST))
            (RETURN (AND (NULL NODOFLG)
                         (APPLY* FN X FILE Z)))))
    (STKNTH -1))))
```

(LISPXPUT

```
(LAMBDA (PROP L ADDFLG LST)
  (PROG (Y)
        (AND (NULL LST)
             (SETQ LST (CAAR LISPXHISTORY)))                        (* |Puts| |property| |at| |top| |level| |of| |entry.|
                                                                    |Used| |mostly| |for| |calls| |with| PROP=ERROR.)
        (COND
           ((SETQ Y (CDR (FMEMB PROP LST)))
            (FRPLACA Y (COND
                          (ADDFLG (NCONC (CAR Y)
                                         L))
                          (T L))))
           (T (NCONC LST (LIST PROP L))))
```

```
          (RETURN L))))

)

(DECLARE\: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \#REDOCNT ARCHIVEFLG ARCHIVEFN ARCHIVELST BOUNDPDUMMY BREAKRESETVALSLST CAR/CDRNIL CHCONLST1
       CLEARSTKLST CLISPARRAY CLISPCHARS CLISPFLG CLISPTRANFLG CONSOLETIME CONSOLETIME0 CPUTIME CPUTIME0
       CTRLUFLG CUTEFLG DISPLAYTERMFLG DWIMFLG EDITHISTORY EDITIME EDITQUIETFLG EDITSTATS EVALQTFORMS FILERDTBL
       FIRSTNAME GREETDATES GREETHIST HISTORYCOMS HISTORYSAVEFN HISTORYSAVEFORMS HISTSTR0 HISTSTR2 HISTSTR3 IT
       LASTHISTORY LISP-RELEASE-VERSION LISPXBUFS LISPXCOMS LISPXFINDSPLST LISPXFNS LISPXHISTORY
       LISPXHISTORYMACROS LISPXMACROS LISPXPRINTFLG LISPXREADFN LISPXSTATS LISPXUSERFN MACSCRATCHSTRING
       NEWUSERFLG P.A.STATS POSTGREETFORMS PREGREETFORMS PRETTYHEADER RANDSTATE READBUFSOURCE REDOCNT REREADFLG
       RESETFORMS SYSFILES TOPLISPXBUFS USERHANDLE USERNAME)
)

(RPAQQ LISP-RELEASE-VERSION 2.0)

(DECLARE\: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY

(BLOCK\: LISPXFINDBLOCK LISPXFIND LISPXFIND0 LISPXFIND1 HISTORYFIND HISTORYFIND1 (ENTRIES LISPXFIND HISTORYFIND)
       (LOCALFREEVARS _FLG
              L LST Z =FLG HISTORYFLG PREDFLG LINE HISTORY TYPE BACKUP QUIETFLG)
       (NOLINKFNS HISTORYMATCH LISPXGETINPUT))

(BLOCK\: NIL ENTRY# EVALQT GETEXPRESSIONFROMEVENTSPEC GREET GREET0 HISTORYMATCH HISTORYSAVE LISPX LISPX/ LISPX/1
       LISPXEVAL LISPXFIND1 LISPXGETINPUT LISPXPRIN1 LISPXPRIN2 LISPXPRINT LISPXPRINTDEF LISPXPRINTDEF0 LISPXPUT
       LISPXREAD LISPXREADBUF LISPXREADP LISPXSPACES LISPXSTOREVALUE LISPXSUBST LISPXTAB LISPXTERPRI
       LISPXTYPEAHEAD LISPXUNREAD LISPXUSE LISPXUSE0 LISPXUSE1 LISPXUSEC PRINTHISTORY PRINTHISTORY1
       PRINTHISTORY2 USEREXEC USERLISPXPRINT VALUEOF VALUOF (LOCALVARS . T)
       (SPECVARS LISPXLINE LISPXID LISPXVALUE LISPXLISTFLG HISTORY ID EVENT BREAKRESETVALS VARS GENLST INITLST
              NAME MESSAGE)
       (LINKFNS . T)
       (NOLINKFNS LISPXTYPEAHEAD UNDOLISPX ARCHIVEFN LISPXFIX LISPXUSE LISPXUSE0 LISPXSUBST LISPXFIND
              HISTORYMATCH PRINTHISTORY DISPLAYTERMP LISPXSTOREVALUE HISTORYSAVEFN ENTEREVALQT PRINTHISTORY1
              PRINTHISTORY2 LISPXFIND HISTORYMATCH LISPXGETINPUT LISPXSUBST ARCHIVEFN LISPXFIX LISPXUSE
              LISPXUSE0 LISPXSUBST HISTORYMATCH PRINTHISTORY DISPLAYTERMP LISPXSTOREVALUE HISTORYSAVEFN
              ENTEREVALQT LISPXTYEAHEAD UNDOLISPX GREETFILENAME))
)

(DECLARE\: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA VALUEOF)

(ADDTOVAR NLAML )

(ADDTOVAR LAMA )
)
```

FUNCTION INDEX

VARIABLE INDEX