File created:	8-Apr-92 21:51:21 **{DSK}<usr>local>lde>lispcore>sources>SEDIT-COMMANDS.;2**
changes to:	(IL:FNS CHANGE-PACKAGE)
	(IL:VARIABLES *POPULAR-PACKAGES*)
	(IL:VARS IL:SEDIT-COMMANDSCOMS)
previous date:	25-Jan-91 14:50:34 {DSK}<usr>local>lde>lispcore>sources>SEDIT-COMMANDS.;1
Read Table:	XCL
Package:	SEDIT
Format:	XCCS

```
(IL:RPAQQ IL:SEDIT-COMMANDSCOMS
          ((IL:PROP IL:FILETYPE IL:SEDIT-COMMANDS)
           (IL:PROP IL:MAKEFILE-ENVIRONMENT IL:SEDIT-COMMANDS)
           (IL:LOCALVARS . T)
           (IL:DECLARE\: IL:DONTCOPY IL:DOEVAL@COMPILE (IL:FILES IL:SEDIT-DECLS))
           (IL:VARIABLES COMMAND-TABLE-SPEC *EDIT-FN* *WRAP-SEARCH* *POPULAR-PACKAGES*)
           (IL:VARS MENU-DESCRIPTION (FIND-CANDIDATE NIL)
                   (SUBSTITUTE-CANDIDATE NIL)
                   (MUTATE-CANDIDATE NIL)
                   (PACKAGE-CANDIDATE NIL)
                   (PRINTBASE-CANDIDATE NIL))
           (IL:INITVARS (CONVERT-UPGRADE 100)
                   (WANT-MENU NIL)
                   (MENUS NIL))
           (IL:CONSTANTS (WORD-DELIM-CHARS (IL:CHARCODE (IL:SPACE IL:CR IL:TAB - IL:{ IL:} IL:[ IL:] IL:\; < >
                                                         IL:\.))))
           (IL:FUNCTIONS

                   ;; pseudo-selections

                   PSEUDO-SELECTION-FROM-SELECTION COMPOSE-PSEUDO-SELECTION DECOMPOSE-PSEUDO-SELECTION
                   SELECTION-FROM-PSEUDO-SELECTION SELECT-PSEUDO-SEGMENT)

           ;; user interface to adding new commands

           (IL:FUNCTIONS ADD-COMMAND GET-SELECTION REPLACE-SELECTION RESET-COMMANDS DEFAULT-COMMANDS)
           (IL:VARIABLES DEFAULT-COMMAND-TABLE-SPEC FIRST-ADD-COMMAND FIRST-ADD-COMMAND-MENU-ENTRY)
           (IL:FUNCTIONS

                   ;; building help menu

                   EQUALIZE-STRING-WIDTHS MINIMUM-STRING-WIDTH MAXIMUM-STRING-WIDTH)
           (IL:FUNCTIONS FIND-AND-DISPLAY-STRUCTURE FIND-AND-DISPLAY-STRUCTURE-BACKWARDS
                   FIND-AND-DISPLAY-SUBSTRUCTURE FIND-AND-DISPLAY-SUBSTRUCTURE-BACKWARDS FIND-NTH-STRUCTURE
                   FIND-NODE-SUBSTRUCTURE FIND-NODE-SUBSTRUCTURE-BACKWARDS FIND-OBJ FIND-SELECTION
                   FIND-SELECTION-BACKWARDS FIND-STRUCTURE FIND-STRUCTURE-BACKWARDS FIND-SUBSTRUCTURE
                   FIND-SUBSTRUCTURE-BACKWARDS GET-USER-STRING SEARCH-OBJ SEARCH-OBJ-BACKWARDS SUBSTITUTE-OBJ
                   SUBSTITUTE-STRUCTURE SUBSTITUTE-SUBSTRUCTURE STRUCTURE-FROM-SELECTION STRUCTURE-FROM-STRING
                   COMMENT-OUT-SELECTION)
           (IL:FNS ADD-MENU BACKSPACE CHANGE-PACKAGE CHANGE-PRINTBASE CHANGE-QUOTE CONVERT-COMMENT
                   CONVERT-COMMENT-STRUCTURE CONVERT-COMMENT-TAIL CREATE-COMMAND-TABLE DEFAULT-EDIT-FN
                   DELETE-SELECTION DELETE-WORD DO-MUTATION EDIT-SELECTION EVAL-SELECTION EXPAND
                   EXTRACT-CURRENT-SELECTION FIND-COMMENT GET-MENU EDIT-HELP HELPMENU INPUT-DOT INPUT-ESCAPE
                   INPUT-NORMAL-CHAR INPUT-QUOTE INPUT-SQUARE-BRACKET INPUT-STRINGDELIM INPUT-TOKENDELIM
                   INSERT-MULTI-ESCAPE INSERT-SPECIAL-CHARACTER INSPECT-SELECTION JOIN MENU-CLOSEFN
                   MENU-FIND-SELECTEDFN MENU-INIT-STATE MENU-PACKAGE-SELECTEDFN MENU-PRINTBASE-SELECTEDFN
                   MENU-SELECTEDFN MENU-SUBSTITUTE-SELECTEDFN MUTATE QUOTE-CURRENT-SELECTION REDISPLAY REDO
                   SELECTED-FN-NAME SKIP-TO-GAP UNDO UNDO-EXTRACT)))

(IL:PUTPROPS IL:SEDIT-COMMANDS IL:FILETYPE :COMPILE-FILE)

(IL:PUTPROPS IL:SEDIT-COMMANDS IL:MAKEFILE-ENVIRONMENT (:READTABLE "XCL" :PACKAGE (DEFPACKAGE IL:SEDIT
                                                                                  (:USE IL:LISP IL:XCL))
                                                        ))

(IL:DECLARE\: IL:DOEVAL@COMPILE IL:DONTCOPY

(IL:LOCALVARS . T)
)

(IL:DECLARE\: IL:DONTCOPY IL:DOEVAL@COMPILE

(IL:FILESLOAD IL:SEDIT-DECLS)
)


(DEFPARAMETER COMMAND-TABLE-SPEC
```

;;; each entry in the COMMAND-TABLE-SPEC should be of the form: (<fn> <help menu entry> <normalize?> <key>+) where <fn> is an atom function
;;; name or a list whose car is the function name and the rest are the extra arguments (beyond context and charcode), <help menu entry> is a list of
;;; strings for the name, key-name, and help-string, <normalize?> is T if the caret should be normalized after this command, and <key>+ is one or more
;;; key specifier which can be passed to charcode (if non-list) or whose car is a termtable syntax (if a list).

'(;; STRUCTURE CONTROL

```
      (INSERT-NULL-LIST NIL T (IL:LEFTPAREN))
      (CLOSE-LIST NIL NIL (IL:RIGHTPAREN))
      (INPUT-SQUARE-BRACKET NIL NIL (IL:LEFTBRACKET)
          (IL:RIGHTBRACKET))
      (INPUT-TOKENDELIM NIL T (IL:SEPRCHAR))
      (INPUT-STRINGDELIM NIL NIL (IL:STRINGDELIM))
      (INPUT-ESCAPE NIL NIL (IL:ESCAPE))
      (INSERT-MULTI-ESCAPE NIL NIL (IL:MULTIPLE-ESCAPE))
      (INSERT-SPECIAL-CHARACTER NIL NIL (IL:PACKAGEDELIM))
      (START-COMMENT NIL NIL ";")
      (INPUT-DOT NIL NIL ".")
      (INSERT-SPECIAL-CHARACTER NIL NIL "#")
      ((INPUT-QUOTE QUOTE)
       NIL NIL "'")
      ((INPUT-QUOTE IL:BQUOTE)
       NIL NIL "`")
      ((INPUT-QUOTE IL:COMMA)
       NIL NIL ",")
      ((INPUT-QUOTE COMMA-AT)
       NIL NIL "@")
      ;; EDIT CONTROL
      (DELETE-SELECTION NIL T IL:DEL)
      (BACKSPACE NIL T IL:BS "^A")
      (DELETE-WORD NIL T "^W")
      ((VERIFY-STRUCTURE NIL T T)
       NIL NIL "^L")
      ((VERIFY-STRUCTURE NIL T NIL)
       NIL NIL "1,^L")
      ;; COMPLETION
      ((COMPLETE :ABORT NIL)
       ("Abort" "M-A" "Complete this edit without installing changes.")
       NIL "1,A" "1,a" (ABORT))
      (NULL ("" "" "")
            NIL 0)
      ((COMPLETE :DONE NIL)
       ("Done" "C-X" "Complete this edit and leave the window open.")
       NIL "^X" (DONE))
      ((COMPLETE :CLOSE)
       ("Done & Close" "C-M-X" "Complete this edit and close the window.")
       NIL "1,^X" (EXIT))
      ((COMPLETE :DONE T)
       ("Done & Compile" "C-C" "Complete this edit, compile, and leave the window open.")
       NIL "^C" (COMPILE))
      ((COMPLETE :CLOSE T)
       ("Done, Compile, & Close" "C-M-C" "Complete this edit, compile, and close the window.")
       NIL "1,^C")
      ;; COMMANDS
      (NULL ("" "" "")
            NIL 0)
      (UNDO ("Undo" "M-U" "Undo the last change made.")
            NIL "1,U" "1,u" 516 (UNDO))
      (REDO ("Redo" "M-R" "Redo the last change undone.")
            NIL "1,R" "1,r" 520 (REDO))
      (NULL ("" "" "")
            NIL 0)
      (FIND-OBJ ("Find" "M-F" "Find the current selection, or prompt for structure to Find.")
            T "1,F" "1,f" 515 (FIND))
      ((FIND-OBJ NIL T)
       ("Reverse Find" "C-M-F" "Find the current selection, or prompt for structure to Find.")
       T "1,^F")
      ((SUBSTITUTE-OBJ NIL NIL T)
       ("Remove" "C-M-S" "Remove structures from within the current selection.")
       NIL "1,^S")
      (SUBSTITUTE-OBJ ("Substitute" "M-S" "Substitute structures within the current selection.")
            NIL "1,S" "1,s" 547 (SUBSTITUTE))
      (SKIP-TO-GAP ("Find Gap" "M-N" "Skip to the next fill in gap.")
            T "1,N" "1,n" 530)
      (NULL ("" "" "")
            NIL 0)
      (EDIT-HELP ("Arglist" "M-H" "Show the argument list for the selected function.")
            NIL "1,H" "1,h" 513 (ARGLIST))
      (CONVERT-COMMENT ("Convert Comment" "M-;" "Convert the old style comments in the current selection.")
            NIL "1,;")
      (COMMENT-OUT-SELECTION NIL NIL "1,^;")
      (EDIT-SELECTION ("Edit" "M-O" "Edit the definition of the current selection.")
            NIL "1,O" "1,o" (EDIT))
      ((EDIT-SELECTION (:CURRENT))
       ("Edit Fast" "C-M-O" "Edit the current definition of the selection.")
       NIL "1,^O")
      (EVAL-SELECTION ("Eval" "M-E" "Evaluate the current selection.")
            NIL "1,E" "1,e" (EVAL))
      (EXPAND ("Expand" "M-X" "Replace the current selection with its definition.")
            NIL "1,X" "1,x" IL:ESC 532 (EXPAND))
      (EXTRACT-CURRENT-SELECTION ("Extract" "M-/" "Extract one level of structure: unquote or unlist.")
```

```
                NIL "1,/" (EXTRACT))
        (INSPECT-SELECTION ("Inspect" "M-I" "Inspect the current selection.")
                NIL "1,I" "1,i" (INSPECT))
        (JOIN ("Join" "M-J" "Join selected items together.")
                NIL "1,J" "1,j" (JOIN))
        (MUTATE ("Mutate" "M-Z" "Prompt for a function to operate on the current selection.")
                NIL "1,Z" "1,z")
      ((PARENTHESIZE-CURRENT-SELECTION NIL)
       ("Parenthesize" "M-(" "Parenthesize the current selection.")
       NIL "1,(" "1,71" (PAREN))
      ((PARENTHESIZE-CURRENT-SELECTION T)
       NIL NIL "1,)" "1,60")
      ((QUOTE-CURRENT-SELECTION QUOTE)
       ("Quote" "M-'" "Quote the current selection.")
       NIL "1,'" (QUOTE))
      ((QUOTE-CURRENT-SELECTION IL:BQUOTE)
       NIL NIL "1,`")
      ((QUOTE-CURRENT-SELECTION IL:COMMA)
       NIL NIL "1,,")
      ((QUOTE-CURRENT-SELECTION COMMA-AT)
       NIL NIL "1,@" "1,62")
      ((QUOTE-CURRENT-SELECTION COMMA-DOT)
       NIL NIL "1,.")
      ((QUOTE-CURRENT-SELECTION FUNCTION)
       NIL NIL "1,#" "1,63")
      (NULL ("" "" "")
                NIL 0)
        (CHANGE-PRINTBASE ("Set Print-Base" "M-B" "Set the print-base for this edit.")
                NIL "1,B" "1,b" (SET-PRINT-BASE))
        (CHANGE-PACKAGE ("Set Package" "M-P" "Set the package to edit in.")
                NIL "1,P" "1,p" (SET-PACKAGE))
        (ADD-MENU ("Attach Menu" "M-M" "Attach a command menu.")
                NIL "1,M" "1,m")
```

;; RANDOM: tells Meta-Space or Meta-Return to scroll to the selection, using the auto-scroller for free.

```
        (TRUE NIL T "1, " "1,
                ")))
```

```
(DEFPARAMETER *EDIT-FN* 'DEFAULT-EDIT-FN)
```

```
(DEFVAR *WRAP-SEARCH* NIL)
```

```
(DEFVAR *POPULAR-PACKAGES* (LIST IL:*INTERLISP-PACKAGE* IL:*LISP-PACKAGE* IL:*COMMON-LISP-PACKAGE*)
                            "SEdit warns if the editing package is not on this list and uses nothing on
                            this list")
```

```
(IL:RPAQQ MENU-DESCRIPTION
        ((IL:PROPS IL:FONT (IL:HELVETICA 10 IL:BOLD))
         ((IL:GROUP (IL:PROPS IL:FORMAT IL:COLUMN IL:COLUMNSPACE 20 IL:ROWSPACE 3)
                ((IL:GROUP (IL:PROPS IL:FORMAT IL:TABLE IL:COLUMNSPACE 12)
                        ((IL:PROPS IL:BOX 1)
                         (IL:LABEL EXIT IL:SELECTEDFN MENU-SELECTEDFN)
                         (IL:LABEL DONE IL:SELECTEDFN MENU-SELECTEDFN)
                         (IL:LABEL ABORT IL:SELECTEDFN MENU-SELECTEDFN IL:MAXWIDTH 39))
                        ((IL:PROPS IL:BOX 1)
                         (IL:LABEL UNDO IL:SELECTEDFN MENU-SELECTEDFN)
                         (IL:LABEL REDO IL:SELECTEDFN MENU-SELECTEDFN)
                         (IL:LABEL ARGLIST IL:SELECTEDFN MENU-SELECTEDFN))))
                ((IL:GROUP (IL:PROPS IL:FORMAT IL:TABLE IL:COLUMNSPACE 12)
                        ((IL:PROPS IL:BOX 1)
                         (IL:LABEL PAREN IL:SELECTEDFN MENU-SELECTEDFN)
                         (IL:LABEL QUOTE IL:SELECTEDFN MENU-SELECTEDFN)
                         (IL:LABEL EXTRACT IL:SELECTEDFN MENU-SELECTEDFN))
                        ((IL:PROPS IL:BOX 1)
                         (IL:LABEL EDIT IL:SELECTEDFN MENU-SELECTEDFN)
                         (IL:LABEL EVAL IL:SELECTEDFN MENU-SELECTEDFN)
                         (IL:LABEL EXPAND IL:SELECTEDFN MENU-SELECTEDFN IL:MAXWIDTH 46))))))
         ((IL:LABEL PRINT-BASE IL:SELECTEDFN MENU-PRINTBASE-SELECTEDFN IL:ID PRINTBASE-ITEM IL:LINKS
                        (IL:EDIT PRINTBASE-VALUE-ITEM))
          (IL:LABEL "" TYPE IL:NUMBER IL:MAXWIDTH 14 IL:ID PRINTBASE-VALUE-ITEM IL:FONT (IL:GACHA 10))
          (IL:LABEL PACKAGE IL:SELECTEDFN MENU-PACKAGE-SELECTEDFN IL:ID PACKAGE-ITEM IL:LINKS (IL:EDIT
                                                                                PACKAGE-NAME-ITEM
                                                                                ))
          (IL:LABEL "" TYPE IL:EDIT IL:ID PACKAGE-NAME-ITEM IL:FONT (IL:GACHA 10)))
         ((IL:GROUP (IL:PROPS IL:FORMAT IL:TABLE)
                ((IL:LABEL FIND\: IL:SELECTEDFN MENU-FIND-SELECTEDFN IL:LINKS (IL:EDIT FINDITEM))
                 (IL:LABEL "" TYPE IL:EDIT IL:ID FINDITEM IL:FONT (IL:GACHA 10)))
                ((IL:LABEL SUBSTITUTE\: IL:SELECTEDFN MENU-SUBSTITUTE-SELECTEDFN IL:LINKS
                        (IL:EDIT SUBSTITUTEITEM FINDITEM FINDITEM))
                 (IL:LABEL "" TYPE IL:EDIT IL:ID SUBSTITUTEITEM IL:FONT (IL:GACHA 10))))))))
```

```
(IL:RPAQQ FIND-CANDIDATE NIL)
```

```
(IL:RPAQQ SUBSTITUTE-CANDIDATE NIL)
```

```
(IL:RPAQQ MUTATE-CANDIDATE NIL)

(IL:RPAQQ PACKAGE-CANDIDATE NIL)

(IL:RPAQQ PRINTBASE-CANDIDATE NIL)

(IL:RPAQ? CONVERT-UPGRADE 100)

(IL:RPAQ? WANT-MENU NIL)

(IL:RPAQ? MENUS NIL)

(IL:DECLARE\: IL:EVAL@COMPILE

(IL:RPAQ WORD-DELIM-CHARS (IL:CHARCODE (IL:SPACE IL:CR IL:TAB - IL:{ IL:} IL:[ IL:] IL:\; < > IL:\.)))

(IL:CONSTANTS (WORD-DELIM-CHARS (IL:CHARCODE (IL:SPACE IL:CR IL:TAB - IL:{ IL:} IL:[ IL:] IL:\; < > IL:\.))))
)
```

(DEFUN **PSEUDO-SELECTION-FROM-SELECTION** (SEL)

;;; A pseudo-selection is either a node or a list of a node and two integers. It's interpreted as the select-node, select-start, and select-end fields of a
;;; selection.

;;; This function takes a selection and creates a pseudo selection from it.

```
   (COMPOSE-PSEUDO-SELECTION (IL:FETCH SELECT-NODE IL:OF SEL)
        (IL:FETCH SELECT-START IL:OF SEL)
        (OR (IL:FETCH SELECT-END IL:OF SEL)
            (IL:FETCH SELECT-START IL:OF SEL))))
```

(DEFUN **COMPOSE-PSEUDO-SELECTION** (NODE &OPTIONAL START END)

;;; A pseudo-selection is either a node or a list of a node and two integers. It's interpreted as the select-node, select-start, and select-end fields of a
;;; selection.

;;; This function takes the fields of a pseudo selection and hands back one.

```
   (COND
      ((LISTP NODE)
       (LIST (IL:FETCH SUPER-NODE IL:OF (FIRST NODE))
           (+ (IL:FETCH SUB-NODE-INDEX IL:OF (FIRST NODE))
              (OR START 0))
           (+ (IL:FETCH SUB-NODE-INDEX IL:OF (FIRST NODE))
              (1- (OR END (LENGTH NODE))))))
      ((OR START END)
       (LIST NODE (OR START END)
           (OR END START)))
      (T NODE)))
```

(DEFUN **DECOMPOSE-PSEUDO-SELECTION** (PSEL)

;;; A pseudo-selection is either a node or a list of a node and two integers. It's interpreted as the select-node, select-start, and select-end fields of a
;;; selection.

;;; This function takes a pseudo selection and hands its fields back as values.

```
   (IF (LISTP PSEL)
       (VALUES (FIRST PSEL)
            (OR (SECOND PSEL)
                (THIRD PSEL))
            (OR (THIRD PSEL)
                (SECOND PSEL)))
       (VALUES PSEL NIL NIL)))
```

(DEFUN **SELECTION-FROM-PSEUDO-SELECTION** (PSEL &OPTIONAL SEL)

;;; A pseudo-selection is either a node or a list of a node and two integers. It's interpreted as the select-node, select-start, and select-end fields of a
;;; selection.

;;; This function takes a pseudo-selection and constructs the corresponding selection. If you don't hand it a selection structure, it conses one.

```
   (UNLESS SEL
       (SETF SEL (IL:CREATE EDIT-SELECTION)))
   (MULTIPLE-VALUE-BIND (NODE START END)
        (DECOMPOSE-PSEUDO-SELECTION PSEL)
     (IL:REPLACE SELECT-NODE IL:OF SEL IL:WITH NODE)
     (IL:REPLACE SELECT-START IL:OF SEL IL:WITH START)
     (IL:REPLACE SELECT-END IL:OF SEL IL:WITH END)
     SEL))
```

```
(DEFUN SELECT-PSEUDO-SEGMENT (CONTEXT PSEL &OPTIONAL SET-POINT? WHERE)
    (MULTIPLE-VALUE-BIND (NODE START END)
        (DECOMPOSE-PSEUDO-SELECTION PSEL)
      (IF START
          (SELECT-NODE-SEGMENT CONTEXT NODE START END)
          (SELECT-NODE CONTEXT NODE SET-POINT? WHERE))))
```

;; user interface to adding new commands

```
(DEFUN ADD-COMMAND (KEY-CODE FORM &OPTIONAL SCROLL? KEY-NAME COMMAND-NAME HELP-STRING)
    (WHEN FIRST-ADD-COMMAND

        ;; cache the command-table-spec so the user can undo this!

        (SETQ DEFAULT-COMMAND-TABLE-SPEC (COPY-TREE COMMAND-TABLE-SPEC))
        (SETQ FIRST-ADD-COMMAND NIL))
    (WHEN (AND KEY-NAME COMMAND-NAME FIRST-ADD-COMMAND-MENU-ENTRY)

        ;; add another separation line to the help menu.

        (NCONC COMMAND-TABLE-SPEC (LIST (LIST 'NULL (LIST "-----" "" "")
                                              NIL 0)))
        (SETQ FIRST-ADD-COMMAND-MENU-ENTRY NIL))
    (NCONC COMMAND-TABLE-SPEC (LIST (LIST FORM (WHEN (AND KEY-NAME COMMAND-NAME)
                                                   (LIST KEY-NAME COMMAND-NAME HELP-STRING))
                                          SCROLL? KEY-CODE)))
    (OR COMMAND-NAME FORM))


(DEFUN GET-SELECTION (CONTEXT)
    (LET ((SELECTION (IL:FETCH SELECTION IL:OF CONTEXT)))
        (COND
           ((OR (NOT (IL:FETCH SELECT-NODE IL:OF SELECTION))
                (NOT (EQ (IL:FETCH SELECT-TYPE IL:OF SELECTION)
                         'STRUCTURE)))
            (VALUES NIL NIL))
           ((IL:FETCH SELECT-START IL:OF SELECTION)
            (VALUES (STRUCTURE-FROM-SELECTION SELECTION)
                    :SUB-LIST))
           (T (VALUES (STRUCTURE-FROM-SELECTION SELECTION)
                      T)))))


(DEFUN REPLACE-SELECTION (CONTEXT STRUCTURE SELECTION-TYPE)
    (UNLESS (OR (EQ SELECTION-TYPE T)
                (EQ SELECTION-TYPE :SUB-LIST))
        (ERROR "Illegal SELECTION-TYPE arg: ~A." SELECTION-TYPE))
    (LET* ((SELECTION (IL:FETCH SELECTION IL:OF CONTEXT))
           (POINT (IL:|fetch| CARET-POINT IL:|of| CONTEXT))
           NEW-NODES)
          (COND
             ((OR (NOT (IL:FETCH SELECT-NODE IL:OF SELECTION))
                  (NOT (EQ (IL:FETCH SELECT-TYPE IL:OF SELECTION)
                           'STRUCTURE)))
              (ERROR "Invalid SEdit selection.  Can't REPLACE-SELECTION."))
             ((EQ SELECTION-TYPE :SUB-LIST)
              (SETQ NEW-NODES (MAPCAR #'(LAMBDA (S)
                                            (PARSE-NEW S CONTEXT))
                                      STRUCTURE)))
             (T (SETQ NEW-NODES (PARSE-NEW STRUCTURE CONTEXT))))
          (PENDING-DELETE POINT SELECTION)
          (INSERT POINT CONTEXT (COPY-LIST NEW-NODES))

          ;; try to select the stuff that was just inserted.

          (SELECT-PSEUDO-SEGMENT CONTEXT (COMPOSE-PSEUDO-SELECTION NEW-NODES))))


(DEFUN RESET-COMMANDS ()
    (LET ((COMMANDS (CREATE-COMMAND-TABLE COMMAND-TABLE-SPEC)))
         (IL:REPLACE (EDIT-ENV COMMAND-TABLE) IL:OF LISP-EDIT-ENVIRONMENT IL:WITH (FIRST COMMANDS))
         (IL:REPLACE (EDIT-ENV HELP-MENU) IL:OF LISP-EDIT-ENVIRONMENT IL:WITH (SECOND COMMANDS)))
    T)


(DEFUN DEFAULT-COMMANDS ()
    (SETQ COMMAND-TABLE-SPEC (COPY-TREE DEFAULT-COMMAND-TABLE-SPEC))
    (SETQ FIRST-ADD-COMMAND-MENU-ENTRY T)
    (RESET-COMMANDS)
    T)


(DEFGLOBALVAR DEFAULT-COMMAND-TABLE-SPEC NIL
    "Used to cache the original command table spec for Reset-Commands")


(DEFGLOBALVAR FIRST-ADD-COMMAND T
    "Used in Add-Command to know if this is the first new command for help-menu update purposes")
```

```
(DEFGLOBALVAR FIRST-ADD-COMMAND-MENU-ENTRY T
    "Used in Add-Command to signal the first time a new command is added to the middle button menu, so that the
    user entries can be separated from the default entries")


(DEFUN EQUALIZE-STRING-WIDTHS (STRING-LIST FONT &OPTIONAL PRIN2? (DESIRED-WIDTH (MAXIMUM-STRING-WIDTH
                                                                                  STRING-LIST FONT PRIN2?))
                                                 (PAD-CHAR #\Space))
```

;;; Increase the width of all the strings in STRING-LIST to DESIRED-WIDTH by padding them on the right with PAD-CHAR.

```
    (DO ((PAD-CHAR-WIDTH (IL:CHARWIDTH (CHAR-CODE PAD-CHAR)
                                       FONT))
         (STR STRING-LIST (REST STR)))
        ((NULL STR)
         STRING-LIST)
      (SETF (FIRST STR)
            (CONCATENATE 'STRING (FIRST STR)
                   (MAKE-STRING (CEILING (- DESIRED-WIDTH (IL:STRINGWIDTH (FIRST STR)
                                                                  FONT PRIN2?))
                                         PAD-CHAR-WIDTH)
                          :INITIAL-ELEMENT PAD-CHAR)))))


(DEFUN MINIMUM-STRING-WIDTH (STRING-LIST FONT PRIN2?)
    (APPLY #'MIN (MAPCAR #'(LAMBDA (S)
                                  (IL:STRINGWIDTH S FONT PRIN2?))
                   STRING-LIST)))


(DEFUN MAXIMUM-STRING-WIDTH (STRING-LIST FONT PRIN2?)
    (APPLY #'MAX (MAPCAR #'(LAMBDA (S)
                                  (IL:STRINGWIDTH S FONT PRIN2?))
                   STRING-LIST)))


(DEFUN FIND-AND-DISPLAY-STRUCTURE (CONTEXT STR &OPTIONAL SCOPE START WRAP?)
```

;;; Find structure and display it by selecting it (point after) and normalizing the selection in the window. SCOPE defaults to the root structure of the
;;; CONTEXT. The WRAP? flag says to wrap failing searches around and try them again (i.e., ignore start and try again).

```
    (LET* ((PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
           (TOP (SUBNODE 1 (IL:FETCH ROOT IL:OF CONTEXT)))
           (TARGET (FIND-STRUCTURE STR (OR SCOPE TOP)
                           START)))
          (COND
            (TARGET (SELECT-NODE CONTEXT TARGET T T)
                    (FORMAT PROMPTWINDOW "~%~S - Found." STR))
            ((AND WRAP? START)
             (FIND-AND-DISPLAY-STRUCTURE CONTEXT STR SCOPE))
            (T (FORMAT PROMPTWINDOW "~%~S - Not found." STR)))))


(DEFUN FIND-AND-DISPLAY-STRUCTURE-BACKWARDS (CONTEXT STR &OPTIONAL SCOPE END WRAP?)
```

;;; Like find-and-display-structure, but searches backwards

```
    (LET* ((PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
           (TOP (SUBNODE 1 (IL:FETCH ROOT IL:OF CONTEXT)))
           (TARGET (FIND-STRUCTURE-BACKWARDS STR (OR SCOPE TOP)
                           END)))
          (COND
            (TARGET (SELECT-NODE CONTEXT TARGET T T)
                    (FORMAT PROMPTWINDOW "~%~S - Found." STR))
            ((AND WRAP? END)
             (FIND-AND-DISPLAY-STRUCTURE-BACKWARDS CONTEXT STR SCOPE))
            (T (FORMAT PROMPTWINDOW "~%~S - Not found." STR)))))


(DEFUN FIND-AND-DISPLAY-SUBSTRUCTURE (CONTEXT STR &OPTIONAL SCOPE START WRAP?)
```

;;; Find substructure and display it by selecting it (pending delete) and normalizing the selection in the window. SCOPE defaults to the root structure of
;;; the CONTEXT. The WRAP? flag says to wrap failing searches around and try them again (i.e., ignore start and try again).

```
    (LET* ((PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
           (TOP (SUBNODE 1 (IL:FETCH ROOT IL:OF CONTEXT)))
           (TARGET (FIND-SUBSTRUCTURE STR (OR SCOPE TOP)
                           START)))
          (COND
            (TARGET (SELECT-PSEUDO-SEGMENT CONTEXT TARGET)
                    (FORMAT PROMPTWINDOW "~%~{~S ~}- Found." STR))
            ((AND WRAP? START)
             (FIND-AND-DISPLAY-SUBSTRUCTURE CONTEXT STR SCOPE))
            (T (FORMAT PROMPTWINDOW "~%~{~S ~}- Not found." STR)))))
```

(DEFUN **FIND-AND-DISPLAY-SUBSTRUCTURE-BACKWARDS** (CONTEXT STR &OPTIONAL SCOPE END WRAP?)

;;; Like find-and-display-substructure but searches backwards

```
    (LET* ((PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
           (TOP (SUBNODE 1 (IL:FETCH ROOT IL:OF CONTEXT)))
           (TARGET (FIND-SUBSTRUCTURE-BACKWARDS STR (OR SCOPE TOP)
                            END)))
          (COND
            (TARGET (SELECT-PSEUDO-SEGMENT CONTEXT TARGET)
                    (FORMAT PROMPTWINDOW "~%~{~S ~}- Found." STR))
            ((AND WRAP? END)
             (FIND-AND-DISPLAY-SUBSTRUCTURE CONTEXT STR SCOPE))
            (T (FORMAT PROMPTWINDOW "~%~{~S ~}- Not found." STR)))))
```

(DEFUN **FIND-NTH-STRUCTURE** (CONTEXT CHARCODE STRUCTURE N)

;;; Find the Nth occurance of Structure in this edit, always starting from the beginning. This function is used as an external command to set the selection
;;; to a desired structure. Find, select, and normalize.

```
    (LET ((TOP (SUBNODE 1 (IL:FETCH ROOT IL:OF CONTEXT))))
        (DO ((M 1 (+ M 1))
             (TARGET (FIND-STRUCTURE STRUCTURE TOP)
                     (FIND-STRUCTURE STRUCTURE TOP (NEXT-NODE TARGET))))
            ((OR (NULL TARGET)
                 (= N M))
             (AND TARGET (SELECT-NODE CONTEXT TARGET T T)))))
      T)
```

(DEFUN **FIND-NODE-SUBSTRUCTURE** (STR STRLEN NODE &OPTIONAL START END CONTINUATION?)

;;; STR is a list of structures of length STRLEN. NODE, together with START and END (which are subnode indices), is taken to indicate a subtree. We
;;; return a pseudo-selection which selects the first sequence of sibling nodes in that subtree whose successive structures match the successive
;;; elements of STR.

;;; "First" here is taken to mean "first in linearization order", so we have to do a careful recursion which: (1a) recursively checks the subtree rooted at the
;;; START subnode of NODE (default the first), (1b) checks if the START subnode starts a matching sibling sequence, (2a) recursively checks the
;;; subtree rooted at the START+1 subnode of NODE, (2b) checks if the START+1 subnode starts a matching sibling sequence, . . ., (Na) recusively
;;; checks the subtree rooted at the END subnode of NODE (default the last), (Nb) checks if the END subnode starts a matching sibling sequence [note
;;; that such a sequence could be only 1 node long since END is the right end of the subtree being checked].

;;; N.B. It might seem that, to get true linearization order, we should check to see if a node starts a matching sibling sequence before we check its
;;; subtree. But since node structures can not be circular, we know that if a match is found in the subtree below a node then that node could not have
;;; started a matching sequence.

;;; The CONTINUATION? flag means that we are continuing a search that has already recursively checked the START subnode, so we skip that
;;; particular recursion. This generally happens when we are working our way up and to the right in some subtree which has already been partially
;;; checked.

```
    (SETF START (OR START 1))
    (LET* ((SUBNODES (IL:FETCH SUB-NODES IL:OF NODE))
           (LASTINDEX (OR END (FIRST SUBNODES))))
          (DO ((SUBS (NTHCDR START SUBNODES)
                     (REST SUBS))
               (INDEX START (1+ INDEX))
               (ENDINDEX (+ START (1- STRLEN))
                         (1+ ENDINDEX))
               (DOSUBS? (NOT CONTINUATION?)
                        T)
               MATCH)
              ((OR (NULL SUBS)
                   (AND END (> INDEX END)))
               NIL)
              (WHEN (AND DOSUBS? (SETF MATCH (FIND-NODE-SUBSTRUCTURE STR STRLEN (FIRST SUBS))))
                    (RETURN MATCH))
              (UNLESS (OR (> ENDINDEX LASTINDEX)
                          (MISMATCH STR SUBS :END2 STRLEN :TEST #'(LAMBDA (S N)
                                                                    (IL:EQUAL S (IL:FETCH STRUCTURE
                                                                                          IL:OF N)))))
                      (RETURN (LIST NODE INDEX ENDINDEX))))))
```

(DEFUN **FIND-NODE-SUBSTRUCTURE-BACKWARDS** (STR STRLEN NODE &OPTIONAL START END CONTINUATION?)

;;; Like find-node-substructure but searches in reverse linearization order.

```
    (LET* ((SUBNODES (IL:FETCH SUB-NODES IL:OF NODE))
           (SUBLENGTH (FIRST SUBNODES)))
          (SETF END (OR END SUBLENGTH))
          (DO ((SUBS (NTHCDR (- SUBLENGTH END)
                             (REVERSE (CDR SUBNODES)))
                     (CDR SUBS))
               (INDEX END (1- INDEX))
               (STARTINDEX (- END (1- STRLEN))
                           (1- STARTINDEX)))
```

```
                (DOSUBS? (NOT CONTINUATION?)
                       T)
                MATCH)
               ((OR (NULL SUBS)
                    (AND START (< INDEX START)))
                NIL)
               (WHEN (AND DOSUBS? (SETF MATCH (FIND-NODE-SUBSTRUCTURE-BACKWARDS STR STRLEN (FIRST SUBS))))
                     (RETURN MATCH))
               (UNLESS (OR (< STARTINDEX 1)
                           (MISMATCH STR SUBS :END2 STRLEN :TEST #'(LAMBDA (S N)
                                                                      (IL:EQUAL S (IL:FETCH STRUCTURE
                                                                                            IL:OF N)))))

                   (RETURN (LIST NODE STARTINDEX INDEX))))))))
```

(DEFUN **FIND-OBJ** (CONTEXT &OPTIONAL CHARCODE FIND-STRING BACKWARDS?)

;;; Find either the passed structure, the selected structure, or a prompted-for structure. The search direction is forward unless BACKWARDS? is
;;; specified.

```
    (CLOSE-OPEN-NODE CONTEXT)
    (LET ((SELECTION (IL:|fetch| SELECTION IL:|of| CONTEXT))
          (WRAP? *WRAP-SEARCH*))
         (COND
            ((AND (NULL FIND-STRING)
                  (IL:|fetch| SELECT-NODE IL:|of| SELECTION)
                  (EQ (IL:|fetch| SELECT-TYPE IL:|of| SELECTION)
                      'STRUCTURE))

             ;; there is a non-string selection

             (IF BACKWARDS?
                 (FIND-SELECTION-BACKWARDS CONTEXT WRAP?)
                 (FIND-SELECTION CONTEXT WRAP?)))
            (T (IF BACKWARDS?
                   (SEARCH-OBJ-BACKWARDS CONTEXT FIND-STRING WRAP?)
                   (SEARCH-OBJ CONTEXT FIND-STRING WRAP?)))))
    T)
```

(DEFUN **FIND-SELECTION** (CONTEXT &OPTIONAL WRAP?)

;;; Find the next match of the current selection and display it.

```
    (LET* ((PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
           (SELECTION (IL:|fetch| SELECTION IL:|of| CONTEXT))
           (NODE (IL:|fetch| SELECT-NODE IL:|of| SELECTION))
           (START (IL:|fetch| SELECT-START IL:|of| SELECTION)))
          (IF START
              ;; a sibling sequence is selected, look for a matching sequence after it

              (FIND-AND-DISPLAY-SUBSTRUCTURE CONTEXT (STRUCTURE-FROM-SELECTION SELECTION)
                     NIL
                     (LIST NODE (1+ START))
                     WRAP?)
              ;; a node is selected, look for a matching node

              (IF (SETF START (NEXT-NODE NODE T))
                  ;; start the search with the following node

                  (FIND-AND-DISPLAY-STRUCTURE CONTEXT (STRUCTURE-FROM-SELECTION SELECTION)
                        NIL START WRAP?)
                  ;; there are no more nodes, either wrap or give up

                  (IF WRAP?
                      (FIND-AND-DISPLAY-STRUCTURE CONTEXT (STRUCTURE-FROM-SELECTION SELECTION))
                      (FORMAT PROMPTWINDOW "~%At end; no more structure to search."))))))
```

(DEFUN **FIND-SELECTION-BACKWARDS** (CONTEXT &OPTIONAL WRAP?)

;;; Find the previous match of the current selection and display it.

```
    (LET* ((PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
           (SELECTION (IL:|fetch| SELECTION IL:|of| CONTEXT))
           (NODE (IL:|fetch| SELECT-NODE IL:|of| SELECTION))
           (END (OR (IL:|fetch| SELECT-START IL:|of| SELECTION)
                    (IL:|fetch| SELECT-END IL:|of| SELECTION))))
          (IF END
              ;; a sibling sequence is selected, look for a matching sequence before it

              (FIND-AND-DISPLAY-SUBSTRUCTURE-BACKWARDS CONTEXT (STRUCTURE-FROM-SELECTION SELECTION)
                     NIL
                     (LIST NODE (1- END))
                     WRAP?)
              ;; a node is selected, look for a matching node

              (IF (SETF END (PREV-NODE NODE T))
```

```
                    ;; start the search with the previous node
                    (FIND-AND-DISPLAY-STRUCTURE-BACKWARDS CONTEXT (STRUCTURE-FROM-SELECTION SELECTION)
                            NIL END WRAP?)
                    ;; there are no more nodes, either wrap or give up
                    (IF WRAP?
                        (FIND-AND-DISPLAY-STRUCTURE-BACKWARDS CONTEXT (STRUCTURE-FROM-SELECTION SELECTION))
                        (FORMAT PROMPTWINDOW "~%At beginning; no more structure to search.")))))))


(DEFUN FIND-STRUCTURE (STR SCOPE &OPTIONAL START)
```

;;; Search forward in linearization order for a node whose structure matches STR. The search is bounded by SCOPE (a pseudo-selection taken to
;;; indicate a subtree) and starts at START (a pseudo-selection taken to indicate its left-most node). START defaults to SCOPE.   The return value is
;;; the first node in SCOPE at or after START whose structure is IL:EQUAL to STR.

;;; N.B. Since node structures can not be circular, no subnode of a node can have structure matching that node. Thus looking for a matching node in
;;; pre-order is the same as looking for one in linearization order. So we do a pre-order search here.

```
    (MULTIPLE-VALUE-BIND (SCOPE-NODE SCOPE-START SCOPE-END)
        (DECOMPOSE-PSEUDO-SELECTION SCOPE)
      (MULTIPLE-VALUE-BIND (START-NODE START-START)
          (DECOMPOSE-PSEUDO-SELECTION START)
        (WHEN (AND (NULL SCOPE-START)
                   (OR (NULL START-NODE)
                       (AND (NULL START-START)
                            (EQ START-NODE SCOPE-NODE)))
                   (IL:EQUAL STR (IL:FETCH STRUCTURE IL:OF SCOPE-NODE)))
```

          ;; special case: the scope includes its root node, we're starting at the root of the scope, and the root of the scope matches the passed
          ;; structure.

```
          (RETURN-FROM FIND-STRUCTURE SCOPE-NODE))
```

        ;; normal case: check all the nodes in the scope subtree in preorder.

```
        (DO* ((MIN-DEPTH (1+ (IL:FETCH DEPTH IL:OF SCOPE-NODE)))
              (NODE (OR (IF START-START
                            (SUBNODE START-START START-NODE)
                            (UNLESS (EQ START-NODE SCOPE-NODE)
                                    START-NODE))
                        (IF SCOPE-START
                            (SUBNODE SCOPE-START SCOPE-NODE)
                            (NEXT-NODE SCOPE-NODE)))
                    (NEXT-NODE NODE)))
             ((OR (NULL NODE)
                  (< (IL:FETCH DEPTH IL:OF NODE)
                     MIN-DEPTH)
                  (AND SCOPE-END (EQ (IL:FETCH SUPER-NODE IL:OF NODE)
                                     SCOPE-NODE)
                               (> (IL:FETCH SUB-NODE-INDEX IL:OF NODE)
                                  SCOPE-END)))
              NIL)
          (WHEN (IL:EQUAL STR (IL:FETCH STRUCTURE IL:OF NODE))
                (RETURN NODE))))))


(DEFUN FIND-STRUCTURE-BACKWARDS (STR SCOPE &OPTIONAL END)
```

;;; like find-structure but searches in reverse linearization order. Actually we search in postorder rather than reverse linearization order but this works
;;; just as well for the same reasons that preorder matches linearization order.

```
    (MULTIPLE-VALUE-BIND (SCOPE-NODE SCOPE-START SCOPE-END)
        (DECOMPOSE-PSEUDO-SELECTION SCOPE)
      (MULTIPLE-VALUE-BIND (END-NODE END-START END-END)
          (DECOMPOSE-PSEUDO-SELECTION END)
        (WHEN (AND (NULL SCOPE-START)
                   (OR (NULL END-NODE)
                       (AND (NULL END-START)
                            (EQ END-NODE SCOPE-NODE)))
                   (IL:EQUAL STR (IL:FETCH STRUCTURE IL:OF SCOPE-NODE)))
```

          ;; special case: the scope includes its root node, we're ending at the root of the scope, and the root of the scope matches the passed
          ;; structure.

```
          (RETURN-FROM FIND-STRUCTURE-BACKWARDS SCOPE-NODE))
```

        ;; normal case: check all the nodes in the scope subtree in postorder.

```
        (DO* ((MIN-DEPTH (1+ (IL:FETCH DEPTH IL:OF SCOPE-NODE)))
              (NODE (OR (IF END-END
                            (SUBNODE END-END END-NODE)
                            (UNLESS (EQ END-NODE SCOPE-NODE)
                                    END-NODE))
                        (IF SCOPE-END
                            (SUBNODE SCOPE-END SCOPE-NODE)
                            (PREV-NODE SCOPE-NODE)))
                    (PREV-NODE NODE)))
             ((OR (NULL NODE)
                  (< (IL:FETCH DEPTH IL:OF NODE)
```

```
                    MIN-DEPTH)
                  (AND SCOPE-START (EQ (IL:FETCH SUPER-NODE IL:OF NODE)
                                       SCOPE-NODE)
                       (< (IL:FETCH SUB-NODE-INDEX IL:OF NODE)
                          SCOPE-START)))
             NIL)
            (WHEN (IL:EQUAL STR (IL:FETCH STRUCTURE IL:OF NODE))
                  (RETURN NODE))))))
```

(DEFUN **FIND-SUBSTRUCTURE** (STR SCOPE &OPTIONAL START)

;;; Search forward in linearization order for a sequence of nodes whose successive structures match the successive elements of STR. The search is
;;; bounded by SCOPE (a pseudo-selection taken to indicate a subtree) and starts at START (a pseudo-selection taken to indicate the left edge of a
;;; subtree). START defaults to SCOPE. The return value is a pseudo-selection indicating the sibling sequence of nodes in SCOPE at or to the right of
;;; START whose successive node structures are IL:EQUAL to the successive members of STR.

;;; N.B. For a sequence of sibling nodes, first in linearization order can not be found by doing a preorder search. See find-node-substructure for details
;;; about the correct search method.

```
    (MULTIPLE-VALUE-BIND (SCOPE-NODE SCOPE-START SCOPE-END)
        (DECOMPOSE-PSEUDO-SELECTION SCOPE)
      (MULTIPLE-VALUE-BIND (START-NODE START-START)
          (DECOMPOSE-PSEUDO-SELECTION START)
        (COND
          ((NULL START-NODE)

           ;; just check the entire scope

           (FIND-NODE-SUBSTRUCTURE STR (LENGTH STR)
                  SCOPE-NODE SCOPE-START SCOPE-END))
          ((EQ START-NODE SCOPE-NODE)

           ;; just check a terminal subtree of the scope

           (FIND-NODE-SUBSTRUCTURE STR (LENGTH STR)
                  SCOPE-NODE START-START SCOPE-END))
          (T ;; check each node from the start subtree up and to the right in the scope subtree.  We carefully resume the recursion that would have
             ;; happened if we had started from the root of the subtree.  This means checking remaining structure in super-nodes on our way from
             ;; the start node back up the subtree.

           (DO ((NODE START-NODE SUPER-NODE)
                (SUPER-NODE (IL:FETCH SUPER-NODE IL:OF START-NODE)
                       (IL:FETCH SUPER-NODE IL:OF NODE))
                (NODE-INDEX (IL:FETCH SUB-NODE-INDEX IL:OF START-NODE)
                       (IL:FETCH SUB-NODE-INDEX IL:OF NODE))
                (CONTINUATION? NIL T)
                (START START-START NODE-INDEX)
                (END NIL (AND (EQ NODE SCOPE-NODE)
                              SCOPE-END))
                (STRLEN (LENGTH STR))
                MATCH)
               ((OR (NULL NODE)
                    (SETF MATCH (FIND-NODE-SUBSTRUCTURE STR STRLEN NODE START END CONTINUATION?))
                    (EQ NODE SCOPE-NODE))
                MATCH)))))))
```

(DEFUN **FIND-SUBSTRUCTURE-BACKWARDS** (STR SCOPE &OPTIONAL END)

;;; Like find-substructure but searches in reverse linearization order.

```
    (MULTIPLE-VALUE-BIND (SCOPE-NODE SCOPE-START SCOPE-END)
        (DECOMPOSE-PSEUDO-SELECTION SCOPE)
      (MULTIPLE-VALUE-BIND (END-NODE END-START END-END)
          (DECOMPOSE-PSEUDO-SELECTION END)
        (COND
          ((NULL END-NODE)

           ;; just check the entire scope

           (FIND-NODE-SUBSTRUCTURE-BACKWARDS STR (LENGTH STR)
                  SCOPE-NODE SCOPE-START SCOPE-END))
          ((EQ END-NODE SCOPE-NODE)

           ;; just check an initial subtree of the scope

           (FIND-NODE-SUBSTRUCTURE-BACKWARDS STR (LENGTH STR)
                  SCOPE-NODE SCOPE-START END-END))
          (T ;; check each node in the initial subtree of scope terminated by the end subtree.  We carefully resume the recursion that would have
             ;; happened if we had started from the root of the scope subtree.  This means checking remaining structure in super-nodes on our way
             ;; from the end node back up the subtree.

           (DO ((NODE END-NODE SUPER-NODE)
                (SUPER-NODE (IL:FETCH SUPER-NODE IL:OF END-NODE)
                       (IL:FETCH SUPER-NODE IL:OF NODE))
                (NODE-INDEX (IL:FETCH SUB-NODE-INDEX IL:OF END-NODE)
                       (IL:FETCH SUB-NODE-INDEX IL:OF NODE))
                (CONTINUATION? NIL T)
                (END END-END NODE-INDEX)
                (START NIL (AND (EQ NODE SCOPE-NODE)
```

```
                                          SCOPE-START))
                        (STRLEN (LENGTH STR))
                        MATCH)
                       ((OR (NULL NODE)
                            (SETF MATCH (FIND-NODE-SUBSTRUCTURE-BACKWARDS STR STRLEN NODE START END CONTINUATION?))
                            (EQ NODE SCOPE-NODE))
                        MATCH)))))))
```

```
(DEFUN GET-USER-STRING (CONTEXT PROMPT DEFAULT)
    (LET ((PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT)))
         (IL:TERPRI PROMPTWINDOW)
         (IL:TTYINPROMPTFORWORD PROMPT DEFAULT NIL PROMPTWINDOW NIL NIL (IL:CHARCODE (IL:CR ^X)))))
```

```
(DEFUN SEARCH-OBJ (CONTEXT &OPTIONAL SEARCH-STRING WRAP?)
```

;;; Search for the the structure(s) in the string SEARCH-OBJ and display them. The search starts just after the current point or selection, if any.

```
    (MULTIPLE-VALUE-BIND (STR STRLEN)
        (STRUCTURE-FROM-STRING (OR SEARCH-STRING (SETF SEARCH-STRING (GET-USER-STRING CONTEXT "Find: "
                                                                       (OR (IL:|fetch| FIND-CANDIDATE
                                                                            IL:|of| CONTEXT)
                                                                          FIND-CANDIDATE)))))
      (COND
        ((< STRLEN 0)
         (FORMAT (GET-PROMPT-WINDOW CONTEXT)
                 " -- Invalid structure.")
         (RETURN-FROM SEARCH-OBJ))
        ((= STRLEN 0)
         (FORMAT (GET-PROMPT-WINDOW CONTEXT)
                 "-- aborted.")
         (RETURN-FROM SEARCH-OBJ)))
```

  ;; update the remembered defaults

```
      (IL:|replace| FIND-CANDIDATE IL:|of| CONTEXT IL:|with| (IL:SETQ FIND-CANDIDATE SEARCH-STRING))
```

  ;; figure out where to search and where to start

```
      (LET* ((SCOPE (SUBNODE 1 (IL:FETCH ROOT IL:OF CONTEXT)))
             (START (LET* ((POINT (IL:|fetch| CARET-POINT IL:|of| CONTEXT))
                           (POINT-TYPE (IL:|fetch| POINT-TYPE IL:|of| POINT))
                           (POINT-NODE (IL:|fetch| POINT-NODE IL:|of| POINT))
                           (POINT-INDEX (IL:|fetch| POINT-INDEX IL:|of| POINT))
                           (SELECTION (IL:|fetch| SELECTION IL:|of| CONTEXT))
                           (SELECT-TYPE (IL:FETCH SELECT-TYPE IL:OF SELECTION))
                           (SELECT-NODE (IL:|fetch| SELECT-NODE IL:|of| SELECTION))
                           (SELECT-START (IL:|fetch| SELECT-START IL:|of| SELECTION)))
                          (COND
                            ((TYPEP POINT-NODE 'EDIT-NODE)
                             (IF (EQ POINT-TYPE 'STRUCTURE)
                                 (NEXT-NODE POINT-NODE POINT-INDEX)
                                 (NEXT-NODE POINT-NODE T)))
                            ((TYPEP SELECT-NODE 'EDIT-NODE)
                             (IF (AND (EQ SELECT-TYPE 'STRUCTURE)
                                      SELECT-START)
                                 (LIST SELECT-NODE (1+ SELECT-START))
                                 (NEXT-NODE SELECT-NODE T)))
                            (T SCOPE)))))
        (UNLESS (OR WRAP? START)
```

            ;; Nothing left to search, and we're not supposed to wrap

```
            (FORMAT (GET-PROMPT-WINDOW CONTEXT)
                    "~%At end; no more structure to search.")
            (RETURN-FROM SEARCH-OBJ))
```

          ;; do the search

```
          (IF (> STRLEN 1)
```

              ;; substructure search

```
              (FIND-AND-DISPLAY-SUBSTRUCTURE CONTEXT STR SCOPE START WRAP?)
```

              ;; structure search

```
              (FIND-AND-DISPLAY-STRUCTURE CONTEXT (FIRST STR)
                    SCOPE START WRAP?)))))
```

```
(DEFUN SEARCH-OBJ-BACKWARDS (CONTEXT &OPTIONAL SEARCH-STRING WRAP?)
```

;;; Like search-obj but searches backwards.

```
    (MULTIPLE-VALUE-BIND (STR STRLEN)
        (STRUCTURE-FROM-STRING (OR SEARCH-STRING (SETF SEARCH-STRING (GET-USER-STRING CONTEXT "Find: "
                                                                       (OR (IL:|fetch| FIND-CANDIDATE
                                                                            IL:|of| CONTEXT)
                                                                          FIND-CANDIDATE)))))
      (COND
        ((< STRLEN 0)
```

```
            (FORMAT (GET-PROMPT-WINDOW CONTEXT)
                    " -- Invalid structure.")
            (RETURN-FROM SEARCH-OBJ-BACKWARDS))
          ((= STRLEN 0)
           (FORMAT (GET-PROMPT-WINDOW CONTEXT)
                   "-- aborted.")
           (RETURN-FROM SEARCH-OBJ-BACKWARDS)))
```

;; update the remembered defaults

(**IL:|replace|** FIND-CANDIDATE **IL:|of|** CONTEXT **IL:|with|** (IL:SETQ FIND-CANDIDATE SEARCH-STRING))

;; figure out where to search and where to start

```
    (LET* ((SCOPE (SUBNODE 1 (IL:FETCH ROOT IL:OF CONTEXT)))
           (END (LET* ((POINT (IL:|fetch| CARET-POINT IL:|of| CONTEXT))
                       (POINT-TYPE (IL:|fetch| POINT-TYPE IL:|of| POINT))
                       (POINT-NODE (IL:|fetch| POINT-NODE IL:|of| POINT))
                       (POINT-INDEX (IL:|fetch| POINT-INDEX IL:|of| POINT))
                       (SELECTION (IL:|fetch| SELECTION IL:|of| CONTEXT))
                       (SELECT-TYPE (IL:FETCH SELECT-TYPE IL:OF SELECTION))
                       (SELECT-NODE (IL:|fetch| SELECT-NODE IL:|of| SELECTION))
                       (SELECT-END (OR (IL:|fetch| SELECT-END IL:|of| SELECTION)
                                       (IL:|fetch| SELECT-START IL:|of| SELECTION))))
                   (COND
                      ((TYPEP POINT-NODE 'EDIT-NODE)
                       (IF (EQ POINT-TYPE 'STRUCTURE)
                           (PREV-NODE POINT-NODE (1+ POINT-INDEX))
                           (PREV-NODE POINT-NODE T)))
                      ((TYPEP SELECT-NODE 'EDIT-NODE)
                       (IF (EQ SELECT-TYPE 'STRUCTURE)
                           (LIST SELECT-NODE (1- SELECT-END))
                           (PREV-NODE SELECT-NODE T)))
                      (T SCOPE)))))
      (UNLESS (OR WRAP? END)
```

;; Nothing left to search, and we're not supposed to wrap

```
        (FORMAT (GET-PROMPT-WINDOW CONTEXT)
                "~%At beginning; no more structure to search.")
        (RETURN-FROM SEARCH-OBJ-BACKWARDS))
```

;; do the search

```
      (IF (> STRLEN 1)
```

;; substructure search

```
          (FIND-AND-DISPLAY-SUBSTRUCTURE-BACKWARDS CONTEXT STR SCOPE END WRAP?)
```

;; structure search

```
          (FIND-AND-DISPLAY-STRUCTURE-BACKWARDS CONTEXT (FIRST STR)
                  SCOPE END WRAP?)))))
```


```
(DEFUN SUBSTITUTE-OBJ (CONTEXT &OPTIONAL CHARCODE OLDSTR NEWSTR REMOVE?)
```

;;; OLDSTR and NEWSTR are strings. In the scope of the selection, replace every occurence of structure matching OLDSTR by structure parsed from
;;; NEWSTR. If REMOVE? is specified, just remove structure matching OLD.

;;; We preserve the selection as best we can. Point gets thrown away.

```
    (CLOSE-OPEN-NODE CONTEXT)
    (LET* ((PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
           (SELECTION (IL:|fetch| SELECTION IL:|of| CONTEXT))
           (SCOPE NIL)
           (TYPE (IF REMOVE?
                     "delet"
                     "substitut")))                                      ; hack!!!
      (UNLESS (AND (IL:|fetch| SELECT-NODE IL:|of| SELECTION)
                   (EQ (IL:|fetch| SELECT-TYPE IL:|of| SELECTION)
                       'STRUCTURE))
         (FORMAT PROMPTWINDOW "~%Please select a structure to ~Ae within." TYPE)
         (RETURN-FROM SUBSTITUTE-OBJ T))
      (SETQ SCOPE (PSEUDO-SELECTION-FROM-SELECTION SELECTION))
      (MULTIPLE-VALUE-BIND (OLD OLDLEN)
          (STRUCTURE-FROM-STRING (OR OLDSTR (SETF OLDSTR (GET-USER-STRING CONTEXT (IF REMOVE?
                                                                              "Delete form: "
                                                                              "Replace old form: ")
                                               (OR (IL:|fetch| FIND-CANDIDATE IL:|of| CONTEXT)
                                                   FIND-CANDIDATE)))))
        (COND
           ((< OLDLEN 0)
            (FORMAT PROMPTWINDOW " -- Invalid structure.")
            (RETURN-FROM SUBSTITUTE-OBJ T))
           ((= OLDLEN 0)
            (FORMAT PROMPTWINDOW "-- aborted.")
            (RETURN-FROM SUBSTITUTE-OBJ T)))
        (MULTIPLE-VALUE-BIND (NEW NEWLEN)
            (IF REMOVE?
                (VALUES NIL 0)
                (STRUCTURE-FROM-STRING (OR NEWSTR (SETF NEWSTR (GET-USER-STRING CONTEXT "with new form: "
```

```
                                                            (OR (IL:|fetch| SUBSTITUTE-CANDIDATE
                                                                 IL:|of| CONTEXT)
                                                             SUBSTITUTE-CANDIDATE))))))
              (COND
                 ((< NEWLEN 0)
                  (FORMAT PROMPTWINDOW " -- Invalid structure.")
                  (RETURN-FROM SUBSTITUTE-OBJ T))
                 ((AND (NOT REMOVE?)
                       (= NEWLEN 0))
                  (FORMAT PROMPTWINDOW "-- aborted.")
                  (RETURN-FROM SUBSTITUTE-OBJ T)))
```

;; update defaults

```
              (IL:|replace| FIND-CANDIDATE IL:|of| CONTEXT IL:|with| (IL:SETQ FIND-CANDIDATE OLDSTR))
              (UNLESS REMOVE?
                     (IL:|replace| SUBSTITUTE-CANDIDATE IL:|of| CONTEXT IL:|with| (IL:SETQ SUBSTITUTE-CANDIDATE NEWSTR)))
```

;; do the substitution, report, and reselect.

```
              (MULTIPLE-VALUE-BIND (NEW-SCOPE SUBCOUNT)
                   (IF (> OLDLEN 1)
                       (SUBSTITUTE-SUBSTRUCTURE CONTEXT OLD NEW SCOPE REMOVE?)
                       (SUBSTITUTE-STRUCTURE CONTEXT (FIRST OLD)
                              NEW SCOPE REMOVE?))
                 (CASE SUBCOUNT
                    (0 (FORMAT PROMPTWINDOW "~%No ~Aions made." TYPE))
                    (1 (FORMAT PROMPTWINDOW "~%1 ~Aion made." TYPE))
                    (OTHERWISE (FORMAT PROMPTWINDOW "~%~A ~Aions made." SUBCOUNT TYPE)))
                 (WHEN NEW-SCOPE (SELECT-PSEUDO-SEGMENT CONTEXT NEW-SCOPE))))))
      T)
```

```
(DEFUN SUBSTITUTE-STRUCTURE (CONTEXT OLD NEW SCOPE &OPTIONAL REMOVE?)
```

;;; Inside SCOPE, replace any node with structure OLD by nodes gotten from parsing NEW. If REMOVE? is given, just delete the old nodes. Returns
;;; two values: the final scope after all substitutions are made, and the number of substitutions/deletions made.

;;; The substitution is done as a single undoable operation, and the current selection and point are thrown away.

```
      (MULTIPLE-VALUE-BIND (SCOPE-NODE SCOPE-START SCOPE-END)
          (DECOMPOSE-PSEUDO-SELECTION SCOPE)
        (LET* ((ROOT (IL:FETCH ROOT IL:OF CONTEXT))                    ; substituting for root is special
               (POINT (IL:FETCH CARET-POINT IL:OF CONTEXT))
               (SELECTION (IL:FETCH SELECTION IL:OF CONTEXT))
               (NEWLEN (IF REMOVE?
                           0
                           (LENGTH NEW)))
               (DELTA-LENGTH (- NEWLEN 1)))
          (START-UNDO-BLOCK)
          (DO* ((TARGET (FIND-STRUCTURE OLD SCOPE)
                        (AND RESUME (FIND-STRUCTURE OLD SCOPE RESUME)))
                (TARGET-SUPER (AND TARGET (IL:FETCH SUPER-NODE IL:OF TARGET))
                        (AND TARGET (IL:FETCH SUPER-NODE IL:OF TARGET)))
                (TARGET-INDEX (AND TARGET (IL:FETCH SUB-NODE-INDEX IL:OF TARGET))
                        (AND TARGET (IL:FETCH SUB-NODE-INDEX IL:OF TARGET)))
                (RESUME (AND TARGET (NEXT-NODE TARGET T))
                        (AND TARGET (NEXT-NODE TARGET T)))
                (NEW-NODES (AND TARGET (NOT REMOVE?)
                                (MAPCAR #'(LAMBDA (S)
                                                  (PARSE-NEW S CONTEXT))
                                        NEW))
                           (AND TARGET (NOT REMOVE?)
                                (MAPCAR #'(LAMBDA (N)
                                                  (COPY-NODE N CONTEXT))
                                        NEW-NODES)))
                (NUMSUBS 0 (1+ NUMSUBS)))
               ((NULL TARGET)
                (END-UNDO-BLOCK)
                (SET-POINT-NOWHERE POINT)
                (SET-SELECTION-NOWHERE SELECTION)
                (VALUES SCOPE NUMSUBS))
```

;; replace the target

```
            (SELECT-NODE CONTEXT TARGET)
            (COND
               (REMOVE? (COND
                            ((EQ TARGET-SUPER ROOT)
```

;; "delete" the root structure by making it nil

```
                             (PENDING-DELETE POINT SELECTION)
                             (INSERT-NULL-LIST CONTEXT))
                            (T (DELETE-SELECTION CONTEXT)))))
               (T (PENDING-DELETE POINT SELECTION)
                  (INSERT POINT CONTEXT (COPY-LIST NEW-NODES)))))
```

;; fix up the scope, if necessary

```
            (COND
               ((EQ TARGET SCOPE-NODE)
```

```
                          ;; matched the scope, so we're done
                          (COND
                             (REMOVE? (SETF SCOPE NIL))
                             ((= NEWLEN 1)
                              (SETF SCOPE (SUBNODE TARGET-INDEX TARGET-SUPER)))
                             (T  ;; replacing the root structure with multiple nodes inserts a new level of list between the root (target-super) and the
                                 ;; multiple nodes inserted.  In this case, make the scope node be the new list node instead of the root itself.

                              (SETF SCOPE (LIST (IF (EQ TARGET-SUPER ROOT)
                                                    (SUBNODE 1 ROOT)
                                                    TARGET-SUPER)
                                                TARGET-INDEX
                                                (+ TARGET-INDEX (1- NEWLEN))))))
                          (SETF RESUME NIL))
                         ((AND SCOPE-START (EQ TARGET-SUPER SCOPE-NODE))

                          ;; matched a direct subnode of an extended scope

                          (WHEN (= TARGET-INDEX SCOPE-END)
                                (SETF RESUME NIL))
                          (SETF (THIRD SCOPE)
                                (INCF SCOPE-END DELTA-LENGTH)))))))))
```

(DEFUN **SUBSTITUTE-SUBSTRUCTURE** (CONTEXT OLD NEW SCOPE &OPTIONAL REMOVE?)

;;; Inside SCOPE, replace any sequences of nodes whose structures sequentially match the elements of OLD by nodes gotten from parsing NEW. If
;;; REMOVE? is given, just delete the old sequences. Returns two values: the final scope after all substitutions are made, and the number of
;;; substitutions/deletions made.

;;; The substitution is done as a single undoable operation, and the current selection and point are thrown away.

```
    (MULTIPLE-VALUE-BIND (SCOPE-NODE SCOPE-START SCOPE-END)
         (DECOMPOSE-PSEUDO-SELECTION SCOPE)
      (LET* ((POINT (IL:|fetch| CARET-POINT IL:|of| CONTEXT))
             (SELECTION (IL:|fetch| SELECTION IL:|of| CONTEXT))
             (NEWLEN (IF REMOVE?
                         0
                         (LENGTH NEW)))
             (DELTA-LENGTH (- NEWLEN (LENGTH OLD))))
        (START-UNDO-BLOCK)
        (DO* ((TARGET (FIND-SUBSTRUCTURE OLD SCOPE)
                      (AND RESUME (FIND-SUBSTRUCTURE OLD SCOPE RESUME)))
              (NEW-NODES (AND TARGET (NOT REMOVE?)
                              (MAPCAR #'(LAMBDA (S)
                                              (PARSE-NEW S CONTEXT))
                                      NEW))
                         (AND TARGET (NOT REMOVE?)
                              (MAPCAR #'(LAMBDA (N)
                                              (COPY-NODE N CONTEXT))
                                      NEW-NODES)))
              (NUMSUBS 0 (1+ NUMSUBS))
              RESUME)
             ((NULL TARGET)
              (END-UNDO-BLOCK)
              (SET-POINT-NOWHERE POINT)
              (SET-SELECTION-NOWHERE SELECTION)
              (VALUES SCOPE NUMSUBS))
          (MULTIPLE-VALUE-BIND (TNODE TSTART TEND)
               (DECOMPOSE-PSEUDO-SELECTION TARGET)

            ;; replace the target

            (SELECT-PSEUDO-SEGMENT CONTEXT TARGET)
            (COND
               (REMOVE? (DELETE-SELECTION CONTEXT))
               (T (PENDING-DELETE POINT SELECTION)
                  (INSERT POINT CONTEXT (COPY-LIST NEW-NODES))))

            ;; fix up the scope, if necessary, and figure where to resume

            (COND
               ((AND SCOPE-START (EQ TNODE SCOPE-NODE))

                ;; matched direct subnodes of an extended scope

                (IF (= TEND SCOPE-END)
                    (SETF RESUME NIL)
                    (SETF RESUME (LIST TNODE (+ TEND 1 DELTA-LENGTH))))
                (SETF (THIRD SCOPE)
                      (INCF SCOPE-END DELTA-LENGTH)))
               (T (SETF RESUME (LIST TNODE (+ TEND 1)))))))))))
```

(DEFUN **STRUCTURE-FROM-SELECTION** (SELECTION)

;;; selection must be a structure selection. Return the structure encompassed by selection, which if the selection is a node is the structure of that node,
;;; and if the selection is a segment a list of the structures of the nodes in that segment.

```
    (LET* ((NODE (IL:FETCH SELECT-NODE IL:OF SELECTION))
```

```
                 (START (IL:FETCH SELECT-START IL:OF SELECTION))
                 (END (OR (IL:FETCH SELECT-END IL:OF SELECTION)
                          START)))
             (COND
                (START (LET ((SUBNODES (IL:FETCH SUB-NODES IL:OF NODE)))
                         (WHEN (<= START END (CAR SUBNODES))
                             (SETF SUBNODES (NTHCDR START SUBNODES))
                             (DO ((STRUCTURE NIL)
                                  (INDEX START (1+ INDEX)))
                                 ((> INDEX END)
                                  (NREVERSE STRUCTURE))
                               (PUSH (IL:FETCH STRUCTURE IL:OF (POP SUBNODES))
                                     STRUCTURE)))))
                (T (IL:FETCH STRUCTURE IL:OF NODE))))))
```

(DEFUN **STRUCTURE-FROM-STRING** (STR)

;;; return all the structures that can be read from string as a list. return a second value saying how many structures there were. If an error is
;;; encountered, a second value of -1 is returned.

```
     (COND
        ((NULL STR)
         (VALUES NIL 0))
        ((STRINGP STR)
         (WITH-INPUT-FROM-STRING (S STR)
             (DO ((RESULTS NIL)
                  (EOF (LIST 'EOF))
                  (COUNT 0 (1+ COUNT))
                  VAL)
                 ((NULL (IL:NLSETQ (SETF VAL (READ S NIL EOF))))
                  (VALUES (NREVERSE RESULTS)
                          -1))
               (IF (EQ VAL EOF)
                   (RETURN (VALUES (NREVERSE RESULTS)
                                   COUNT))
                   (PUSH VAL RESULTS)))))
        (T (VALUES NIL -1))))
```

(DEFUN **COMMENT-OUT-SELECTION** (CONTEXT CHARCODE)

;;; given a sequence of whole structure selections, build a 5 level comment node and replace the nodes with the comment.

```
     (LET* ((SELECTION (IL:FETCH SELECTION IL:OF CONTEXT))
            (POINT (IL:FETCH CARET-POINT IL:OF CONTEXT))
            (NODE (IL:FETCH SELECT-NODE IL:OF SELECTION))
            (START (IL:FETCH SELECT-START IL:OF SELECTION))
            (STR (COND
                    ((OR (NULL NODE)
                         (NOT (EQ (IL:FETCH SELECT-TYPE IL:OF SELECTION)
                                  'STRUCTURE)))
                     (FORMAT (GET-PROMPT-WINDOW CONTEXT)
                             "~%Select whole structure or structures to comment out.")
                     NIL)
                    (START (WITH-OUTPUT-TO-STRING (S)
                              (IL:BIND BLANK-BEFORE IL:FOR I IL:FROM START
                                 IL:TO (OR (IL:FETCH SELECT-END IL:OF SELECTION)
                                           START)
                                 IL:AS X IL:ON (CDR (IL:NTH (IL:FETCH SUB-NODES IL:OF NODE)
                                                            START))
                                 IL:DO (IF BLANK-BEFORE
                                           (WRITE-CHAR #\Space S)
                                           (SETQ BLANK-BEFORE T))
                                       (PRIN1 (IL:FETCH STRUCTURE IL:OF (CAR X))
                                              S))))
                    (T (FORMAT NIL "~S" (IL:FETCH STRUCTURE IL:OF NODE)))))))
        (WHEN STR
            (LET ((NEW-NODE (PARSE-NEW (LIST 'IL:* 'IL:\| STR)
                                       CONTEXT)))
               (START-UNDO-BLOCK)
               (DELETE-SELECTION CONTEXT)
               (INSERT POINT CONTEXT NEW-NODE)
               (SELECT-NODE CONTEXT NEW-NODE)
               (IL:REPLACE PENDING-DELETE? IL:OF SELECTION IL:WITH NIL)
               (END-UNDO-BLOCK))))
     T)
```

(IL:DEFINEQ

(**ADD-MENU**
  (IL:LAMBDA (CONTEXT)   ; Edited 7-Jul-87 09:27 by DCB
    (LET ((WINDOW (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT))
          (PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
          MENU)
         (COND
            ((IL:WINDOWPROP WINDOW 'MENU)

```
              (IL:|printout| PROMPTWINDOW T "This SEdit already has a menu."))
          (T (IL:|printout| PROMPTWINDOW T "Creating menu...")
            (IL:SETQ MENU (GET-MENU CONTEXT))
            (IL:ATTACHWINDOW MENU WINDOW NIL NIL 'IL:LOCALCLOSE)
            (IL:WINDOWPROP MENU 'IL:REJECTMAINCOMS '(IL:SHAPEW))
            (IL:WINDOWADDPROP WINDOW 'IL:RESHAPEFN 'IL:REPOSITIONATTACHEDWINDOWS)
            (IL:WINDOWPROP WINDOW 'MENU MENU)
            (IL:TERPRI PROMPTWINDOW))))
      T))
```

(**BACKSPACE**
   (IL:LAMBDA (CONTEXT)   ; Edited 7-Jul-87 09:27 by DCB

;;; implements the backspace key. if there's a caret, find the appropriate method for the node it's in. the type methods must take care of any selection
;;; as appropriate. If there's a pending delete selection, consider backspace an undefined operation and punt, unless it's a quoted gap, let quote deal
;;; with it.

```
      (LET* ((POINT (IL:FETCH CARET-POINT IL:OF CONTEXT))
             (NODE (IL:FETCH POINT-NODE IL:OF POINT)))
            (COND
              ((IL:TYPE? EDIT-NODE NODE)
               (FUNCALL (IL:FETCH BACK-SPACE IL:OF (IL:FETCH NODE-TYPE IL:OF NODE))
                        NODE CONTEXT (IL:FETCH POINT-INDEX IL:OF POINT)
                        (IL:FETCH POINT-STRING IL:OF POINT)))
              (NODE (LET* ((SELECTION NODE))
                         (WHEN (AND (IL:SETQ NODE (IL:FETCH SELECT-NODE IL:OF SELECTION))
                                    (EQ TYPE-GAP (IL:FETCH NODE-TYPE IL:OF NODE))
                                    (EQ TYPE-QUOTE (IL:FETCH NODE-TYPE IL:OF (IL:FETCH SUPER-NODE IL:OF NODE))))
                              (BACKSPACE-QUOTE (IL:FETCH SUPER-NODE IL:OF NODE)
                                               CONTEXT T))))))
      T))
```

(**CHANGE-PACKAGE**
   (IL:LAMBDA (CONTEXT CHARCODE NEW-PACKAGE NEW-PACKAGE-NAME)   ; Edited 5-Dec-90 14:19 by woz

;;; new.package and new.package.name will be set if coming from the menu. the menu selectedfn already checked valid package. otherwise coming
;;; from the keyboard, and need to prompt.

```
      (CLOSE-OPEN-NODE CONTEXT)
      (LET ((PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
            (WINDOW (IL:|fetch| DISPLAY-WINDOW IL:|of| CONTEXT)))
           (WHEN (NULL NEW-PACKAGE)
             (IL:TERPRI PROMPTWINDOW)
             (IL:SETQ NEW-PACKAGE-NAME (IL:U-CASE (IL:TTYINPROMPTFORWORD "New package: " PACKAGE-CANDIDATE NIL
                                                    PROMPTWINDOW NIL NIL (IL:CHARCODE (IL:CR ^X)))))
```
             ;; if have input then look for package, and if found reset candidate to full package name (user could have typed it abbreviation) of new
             ;; package.
```
             (IF (IL:STRINGP NEW-PACKAGE-NAME)
                 (IL:SETQ NEW-PACKAGE (FIND-PACKAGE NEW-PACKAGE-NAME)))
             (IF NEW-PACKAGE
                 (IL:SETQ PACKAGE-CANDIDATE (IL:SETQ NEW-PACKAGE-NAME (PACKAGE-NAME NEW-PACKAGE)))))
           (COND
             ((EQ NEW-PACKAGE *PACKAGE*)
              (FORMAT PROMPTWINDOW "~%Already editing in package ~A." NEW-PACKAGE-NAME))
             (NEW-PACKAGE (IL:SETQ *PACKAGE* NEW-PACKAGE)
                  (SAVE-PROFILE (IL:|fetch| PROFILE IL:|of| CONTEXT))
                  (VERIFY-STRUCTURE CONTEXT NIL NIL T)
                  (WHEN (IL:WINDOWPROP WINDOW 'MENU)
                    (IL:FM.CHANGELABEL 'PACKAGE-NAME-ITEM NEW-PACKAGE-NAME (IL:WINDOWPROP WINDOW 'MENU)))
                  (FORMAT PROMPTWINDOW "~%Now editing in package ~A" NEW-PACKAGE-NAME)
                  (IF (AND (NOT (MEMBER NEW-PACKAGE *POPULAR-PACKAGES*))
                           (NOT (INTERSECTION *POPULAR-PACKAGES* (PACKAGE-USE-LIST NEW-PACKAGE))))
                      (FORMAT PROMPTWINDOW " (which uses no popular packages).")
                      (FORMAT PROMPTWINDOW ".")))
             (NEW-PACKAGE-NAME (IL:|printout| PROMPTWINDOW T "No such package: " NEW-PACKAGE-NAME))
             (T (IL:|printout| PROMPTWINDOW "...aborted"))))
      T))
```

(**CHANGE-PRINTBASE**
   (IL:LAMBDA (CONTEXT CHARCODE NEW-PRINTBASE)   ; Edited 5-Dec-90 14:18 by woz

;;; new.printbase will be set (and valid) if coming from the menu. otherwise, prompt.

```
      (CLOSE-OPEN-NODE CONTEXT)
      (LET ((PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
            NEW-PRINTBASE-STRING)
           (WHEN (NULL NEW-PRINTBASE)
             (IL:TERPRI PROMPTWINDOW)
             (IL:SETQ NEW-PRINTBASE-STRING (IL:TTYINPROMPTFORWORD "New print-base: " PRINTBASE-CANDIDATE NIL
                                               PROMPTWINDOW NIL NIL (IL:CHARCODE (IL:CR ^X))))
             (OR (AND (IL:STRINGP NEW-PRINTBASE-STRING)
                      (IL:SETQ NEW-PRINTBASE (IL:FIXP (CAR (IL:NLSETQ (IL:READ (IL:OPENSTRINGSTREAM
                                                                      NEW-PRINTBASE-STRING
```

```
                                                                        'IL:INPUT))))))
                    (IL:IGREATERP NEW-PRINTBASE 1)
                    (IL:ILEQ NEW-PRINTBASE 36)
                    (IL:SETQ PRINTBASE-CANDIDATE NEW-PRINTBASE-STRING))
               (IL:SETQ NEW-PRINTBASE NIL)))
          (COND
             (NEW-PRINTBASE (IL:SETQ *PRINT-BASE* NEW-PRINTBASE)
                    (IL:SETQ *PRINT-RADIX* (IL:NEQ *PRINT-BASE* 10))
                    (SAVE-PROFILE (IL:|fetch| PROFILE IL:|of| CONTEXT))
                    (VERIFY-STRUCTURE CONTEXT NIL NIL NIL T)
                    (WHEN (IL:WINDOWPROP (IL:|fetch| DISPLAY-WINDOW IL:|of| CONTEXT)
                                  'MENU)
                        (LET ((*PRINT-BASE* 10))                    ; make display be in base 10
                              (IL:FM.CHANGESTATE 'PRINTBASE-VALUE-ITEM NEW-PRINTBASE (IL:WINDOWPROP (IL:|fetch|
                                                                                            DISPLAY-WINDOW
                                                                                          IL:|of|
                                                                                              CONTEXT
                                                                                              )
                                                                                      'MENU)))))
             (T (IL:|printout| PROMPTWINDOW T "Illegal print-base: " NEW-PRINTBASE-STRING)))))
      T))
```

₍**CHANGE-QUOTE**
```
  (IL:LAMBDA (QUOTE-NODE CONTEXT QUOTE-TYPE)                          ; Edited  7-Jul-87 09:27 by DCB
      (RPLACA (IL:FETCH STRUCTURE IL:OF QUOTE-NODE)
              (QUOTE-WRAPPER QUOTE-TYPE))
      (IL:REPLACE UNASSIGNED IL:OF QUOTE-NODE IL:WITH (IL:LISTGET (IL:FETCH QUOTE-STRING
                                                                    IL:OF (IL:FETCH ENVIRONMENT IL:OF CONTEXT))
                                                          QUOTE-TYPE))
      (NOTE-CHANGE QUOTE-NODE CONTEXT)))
```

₍**CONVERT-COMMENT**
```
  (IL:LAMBDA (CONTEXT)                                               ; Edited  7-Jul-87 09:27 by DCB
      (LET* ((PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
             (SELECTION (IL:FETCH SELECTION IL:OF CONTEXT))
             (POINT (IL:FETCH CARET-POINT IL:OF CONTEXT))
             (NODE (IL:FETCH SELECT-NODE IL:OF SELECTION))
             (START (IL:FETCH SELECT-START IL:OF SELECTION))
             (NUMBER-OF-COMMENTS 0)
             SELECT-END)
            (COND
               ((AND NODE (EQ (IL:FETCH SELECT-TYPE IL:OF SELECTION)
                              'STRUCTURE))                           ; there is a selection to substitute within
                   (COND
                      (START (IL:SETQ NODE (SUBNODE START NODE))
                             (IL:SETQ SELECT-END (OR (IL:FETCH SELECT-END IL:OF SELECTION)
                                                     START)))
                      (T (IL:SETQ SELECT-END (IL:FETCH SUB-NODE-INDEX IL:OF NODE))))
                   (START-UNDO-BLOCK)
                   (IL:BIND (NEXT-NODE IL:_ NODE)
                         (DEPTH IL:_ (IL:FETCH DEPTH IL:OF NODE))
                         NEW-NODE IL:WHILE (IL:SETQ NODE (FIND-COMMENT NEXT-NODE CONTEXT DEPTH SELECT-END))
                      IL:DO   ;; move past it so we're not  pointing to a dead node after the substitution

                         (IL:SETQ NEXT-NODE (NEXT-NODE NODE T))
                         (WHEN (NOT (IL:FMEMB (CADR (IL:FETCH STRUCTURE IL:OF NODE))
                                        COMMENT-MARKERS))

                             ;; this is an old comment.  convert it

                             (IL:SETQ NEW-NODE (PARSE-NEW (CONVERT-COMMENT-STRUCTURE (IL:FETCH STRUCTURE
                                                                                         IL:OF NODE))
                                                    CONTEXT))
                             (REPLACE-NODE CONTEXT NODE NEW-NODE)
                             (IL:ADD NUMBER-OF-COMMENTS 1))

                         ;; and continue the search

                         )
                   (END-UNDO-BLOCK)
                   (IL:|printout| PROMPTWINDOW T (IF (EQ 0 NUMBER-OF-COMMENTS)
                                                     "No"
                                                     NUMBER-OF-COMMENTS)
                        (IF (EQ NUMBER-OF-COMMENTS 1)
                            " comment converted."
                            " comments converted."))
                   ;; finally reset the point

                   (WHEN (NOT (EQ 0 NUMBER-OF-COMMENTS))
                        (SET-POINT-NOWHERE POINT)
                        (IL:REPLACE PENDING-DELETE? IL:OF SELECTION IL:WITH NIL)))
               (T (IL:|printout| PROMPTWINDOW T "Select structure to convert comments within.")))))
      T))
```

₍**CONVERT-COMMENT-STRUCTURE**
```
  (IL:LAMBDA (EXPR)                                                  ; Edited 17-Jul-87 09:48 by DCB
```

```
      (LET (2-STARS COMTAIL COMCHAR)
           (COND
              ((AND (IL:EQMEMB (CAR EXPR)
                           IL:COMMENTFLG)
                    (IL:LISTP (CDR EXPR))
                    (NOT (IL:FMEMB (CADR EXPR)
                               '(IL:E IL:DECLARATIONS\: IL:CLISP\:)))
                    (IL:LISTP (IL:SETQ COMTAIL (IF (IL:SETQ 2-STARS (IL:EQMEMB (CADR EXPR)
                                                                              IL:COMMENTFLG))
                                                   (CDDR EXPR)
                                                   (CDR EXPR)))))
               (IL:SETQ COMCHAR (OR (CAR (IL:LISTP IL:COMMENTFLG))
                                    IL:COMMENTFLG))
               (COND
                  ((AND (IL:NLISTP (CDR COMTAIL))
                        (IL:STRINGP (CAR COMTAIL)))                      ; already stringified.  now semicolonize
                   (COND
                      (2-STARS (IL:PUSH COMTAIL LEVEL-3-COMMENT))
                      ((IL:IGEQ (IL:NCHARS (CAR COMTAIL))
                                CONVERT-UPGRADE)
                       (IL:PUSH COMTAIL LEVEL-2-COMMENT))
                      (T (IL:PUSH COMTAIL LEVEL-1-COMMENT)))
                   (CONS COMCHAR COMTAIL))
                  ((AND (IL:NLISTP (CDDR COMTAIL))
                        (IL:STRINGP (CADR COMTAIL)))                     ; could be an edit date
                   EXPR)
                  (T                                                      ;; COMTAIL is where the comment starts, and this is not a funny evaluated comment.
                     (IL:SETQ COMTAIL (LIST (IL:CONCATLIST (CONVERT-COMMENT-TAIL COMTAIL (CONS)))))
                     (COND
                        (2-STARS (IL:PUSH COMTAIL LEVEL-3-COMMENT))
                        ((IL:IGEQ (IL:NCHARS (CAR COMTAIL))
                                  CONVERT-UPGRADE)
                         (IL:PUSH COMTAIL LEVEL-2-COMMENT))
                        (T (IL:PUSH COMTAIL LEVEL-1-COMMENT)))
                     (CONS COMCHAR COMTAIL))))
              (T                                                          ; Not convertible
                 EXPR)))))
```

(**CONVERT-COMMENT-TAIL**
```
  (IL:LAMBDA (TAIL STREAM)                                              ; Edited 17-Jul-87 09:49 by DCB
```

;;;; to remove the dependency on WITH-OUTPUT-TO-STRING, which probably isn't very efficient and isn't available in koto anyway, we instead just
;;;; accumulate a list of strings, and concatlist them at the end. STREAM should be a TCONC pointer

```
    (IL:WHILE TAIL IL:BIND IL:X NSPACES IL:DO (IL:SETQ NSPACES 1)
                                              (COND
                                                 ((IL:NLISTP TAIL) ; Dotted tail of some super list
                                                  (IL:TCONC STREAM " . ")
                                                  (IL:SETQ IL:X TAIL)
                                                  (IL:SETQ TAIL NIL))
                                                 (T (IL:SETQ IL:X (CAR TAIL))
                                                    (IL:SETQ TAIL (CDR TAIL))))
                                              (COND
                                                 ((IL:STRINGP IL:X)
                                                                        ; Turn quote marks into single quotes
                                                  (IL:LCONC STREAM (LIST "'" IL:X "'")))
                                                 ((IL:LISTP IL:X)
                                                  (IL:TCONC STREAM "(")
                                                  (COND
                                                     ((EQ (CAR IL:X)
                                                          '-)           ; Suppress line break that would occur here:  MAKE IT A BIG
                                                                        ; DASH
                                                      (IL:TCONC STREAM (IF (CDR IL:X)
                                                                           "--- "
                                                                           "---"))
                                                      (IL:POP IL:X)))
                                                  (CONVERT-COMMENT-TAIL IL:X STREAM)
                                                  (IL:TCONC STREAM ")")
                                                  (IL:SELECTQ (CAR (IL:LISTP TAIL))
                                                       ((IL:\. IL:\, IL:\; IL:?)
                                                        (IL:SETQ NSPACES 0))
                                                       NIL))
                                                 ((EQ IL:X '-)          ; old style "force line break": MAKE IT A BIG DASH
                                                  (IL:TCONC STREAM "---"))
                                                 (T (IL:TCONC STREAM IL:X)
                                                    (IL:SELCHARQ (IL:NTHCHARCODE IL:X -1)
                                                         ((IL:\. IL:\; IL:?)
                                                              (IL:SETQ NSPACES 2))
                                                         NIL)))
                                              (COND
                                                 ((AND (IL:NEQ NSPACES 0)
                                                       TAIL)
                                                  (IL:TCONC STREAM (IF (EQ NSPACES 1)
                                                                       " "
                                                                       "  ")))))
```

```
        (CAR STREAM)))
```

(**CREATE-COMMAND-TABLE**
```
   (IL:LAMBDA (DESCRIPTION)                                          ; Edited 13-Jun-88 19:02 by Snow
```

;;; each entry in the COMMAND-TABLE-SPEC should be of the form: (<fn> <help menu entry> <normalize?> <key>+) where <fn> is an atom function
;;; name or a list whose car is the function name and the rest are the extra arguments (beyond context and charcode), <help menu entry> is a list of
;;; strings for the name, key-name, and help-string, <normalize?> is T if the caret should be normalized after this command, and <key>+ is one or more
;;; key specifier which can be passed to charcode (if non-list) or whose car is a termtable syntax (if a list).

```
        (LET ((TABLE (MAKE-HASH-TABLE :SIZE 95 :REHASH-SIZE 5))
              (MENU-ITEMS NIL)
              (MENU-LEFT NIL)
              (MENU-RIGHT NIL)
              FN ENTRY)
            (IL:|for| COMMAND IL:|in| DESCRIPTION IL:|do|
```
                                        ;; get fn for this command. The first thing in COMMAND is either an atom (a simple function name),
                                        ;; or a list of the form (<fn-name> <extra-args>*). Make a "command form" for sedit of the form
                                        ;; (<fn-name> <normalize?> <extra-args>*)
```
                                     (SETQ FN (IF (CONSP (SETQ ENTRY (FIRST COMMAND)))
                                                  (LIST* (FIRST ENTRY)
                                                         (THIRD COMMAND)
                                                         (REST ENTRY))
                                                  (LIST ENTRY (THIRD COMMAND))))
```
                                        ;; check for help menu entry: save left and right columns for tabulating later, and collect the menu
                                        ;; items, without the label, but with the selectedfn and the help string.
```
                                     (WHEN (IL:SETQ ENTRY (SECOND COMMAND))
                                         (PUSH (FIRST ENTRY)
                                               MENU-LEFT)
                                         (PUSH (SECOND ENTRY)
                                               MENU-RIGHT)
                                         (PUSH (LIST (IL:KWOTE FN)
                                                     (THIRD ENTRY))
                                               MENU-ITEMS))
```
                                        ;; for each of the keys for this command, make a table entry. if key is a list, use the symbol in it to key
                                        ;; on (for syntax and attached menu entries), else treat it as a charcode spec.
```
                                     (IL:|for| KEY IL:|in| (CDDDR COMMAND)
                                        IL:|do| (SETF (GETHASH (IF (IL:LISTP KEY)
                                                                   (CAR KEY)
                                                                   (CHARCODE KEY))
                                                               TABLE)
                                                      FN)))
```
            ;; return list of command table and help menu items
```
            (LIST TABLE (LIST MENU-ITEMS MENU-LEFT MENU-RIGHT)))))
```

(**DEFAULT-EDIT-FN**
```
   (IL:LAMBDA (OBJ OPTIONS)                                          ; Edited  5-Jul-88 15:12 by woz
      (ED OBJ (LIST* :DISPLAY :DONTWAIT OPTIONS))))
```

(**DELETE-SELECTION**
```
   (IL:LAMBDA (CONTEXT)                                              ; Edited  7-Jul-87 09:27 by DCB
```
      ;; delete the currently selected nodes, and set the caret point to where they were.
```
      (LET ((SELECTION (IL:FETCH SELECTION IL:OF CONTEXT)))
           (AND (IL:FETCH SELECT-NODE IL:OF SELECTION)
                (DELETE-NODES (IL:FETCH SELECT-NODE IL:OF SELECTION)
                              CONTEXT
                              (IL:FETCH SELECT-START IL:OF SELECTION)
                              (IL:FETCH SELECT-END IL:OF SELECTION)
                              (IL:FETCH CARET-POINT IL:OF CONTEXT)
                              (IL:FETCH SELECT-STRING IL:OF SELECTION)))
           (WHEN (NOT (AND (IL:FETCH SELECT-NODE IL:OF SELECTION)
                           (EQ TYPE-GAP (IL:FETCH NODE-TYPE IL:OF (IL:FETCH SELECT-NODE IL:OF SELECTION)))))
                 (SET-SELECTION-NOWHERE SELECTION)))
      T))
```

(**DELETE-WORD**
```
   (IL:LAMBDA (CONTEXT)                                              ; Edited 24-Nov-87 10:02 by DCB
      (CLOSE-OPEN-NODE CONTEXT)
      (LET* ((POINT (IL:FETCH CARET-POINT IL:OF CONTEXT))
             (SELECTION (IL:FETCH SELECTION IL:OF CONTEXT))
             (NODE (IL:FETCH POINT-NODE IL:OF POINT))
             (END (IL:FETCH POINT-INDEX IL:OF POINT))
             (STRING (IL:FETCH POINT-STRING IL:OF POINT))
             START)
```
            ;; don't do anything if there's no point or a pending delete selection.
```
            (WHEN (AND NODE (OR (NOT (IL:FETCH SELECT-NODE IL:OF SELECTION))
                                (NOT (IL:FETCH PENDING-DELETE? IL:OF SELECTION))))
```

```
                (IL:SELECTQ (IL:FETCH POINT-TYPE IL:OF POINT)
                        (ATOM (DELETE-NODES NODE CONTEXT 1 END POINT STRING))
                        (ESC-ATOM (DELETE-NODES NODE CONTEXT 1 END POINT STRING))
                        (STRING (COND
                                    ((EQ (IL:FETCH NODE-TYPE IL:OF NODE)
                                         TYPE-COMMENT)
                                     (MAP-COMMENT-INDEX CONTEXT NODE END)
                                     (COND
                                        ((IL:IGREATERP END 0)
                                         (DELETE-NODES NODE CONTEXT (IL:IDIFFERENCE (IL:ADD1 END)
                                                                                   (IL:FETCH \\x IL:OF CONTEXT))
                                                       END POINT STRING))
                                        ((NULL (CDR (IL:FETCH SUB-NODES IL:OF NODE)))
                                         (DELETE-NODES NODE CONTEXT NIL NIL POINT STRING))))
                                    (T (IL:SETQ START END)
                                       (COND
                                          ((IL:IGREATERP START 0)        ; backup over preceding whitespace
                                           (IL:WHILE (AND (IL:NEQ START 1)
                                                          (IL:FMEMB (IL:NTHCHARCODE STRING START)
                                                                    WORD-DELIM-CHARS))
                                              IL:DO (IL:SETQ START (IL:SUB1 START)))
                                                                         ; backup over preceding word
                                           (IL:UNTIL (OR (EQ START 0)
                                                         (IL:FMEMB (IL:NTHCHARCODE STRING START)
                                                                   WORD-DELIM-CHARS))
                                              IL:DO (IL:SETQ START (IL:SUB1 START)))
                                           (DELETE-NODES NODE CONTEXT (IL:ADD1 START)
                                                         END POINT STRING))
                                          ((EQ 0 (IL:NCHARS STRING))
                                           (DELETE-NODES NODE CONTEXT NIL NIL POINT STRING))))))
                        (STRUCTURE (COND
                                       ((IL:IGREATERP END 0)
                                        (DELETE-NODES NODE CONTEXT END NIL POINT STRING))
                                       ((NULL (CDR (IL:FETCH SUB-NODES IL:OF NODE)))
                                        (DELETE-NODES NODE CONTEXT NIL NIL POINT STRING))))
                        NIL)
                    (WHEN (NOT (AND (IL:FETCH SELECT-NODE IL:OF SELECTION)
                                    (EQ TYPE-GAP (IL:FETCH NODE-TYPE IL:OF (IL:FETCH SELECT-NODE IL:OF SELECTION)))))
                      ;; cancel the selection unless its pending delete (ctrl-w doesn't do anything) or its a gap, which could have been created by the
                      ;; deletion.
                      (SET-SELECTION-NOWHERE SELECTION)))))
        T))
```

(**DO-MUTATION**
```
  (IL:LAMBDA (CONTEXT NODE MUTATOR)                                              ; Edited  7-Jul-87 09:27 by DCB
```

;;; this guy actually applies the mutation and replaces the sedit structure. should return T if okay, and NIL if error occured durng mutation.

```
    (LET ((RESULT (IL:NLSETQ (FUNCALL MUTATOR (IL:FETCH STRUCTURE IL:OF NODE)))))
        (WHEN RESULT
            ;; assume result is not equal to node's Structure.  otherwise, why would mutate have been called?
            (REPLACE-NODE CONTEXT NODE (PARSE-NEW (CAR RESULT)
                                                 CONTEXT))          ; return T
          T))))
```

(**EDIT-SELECTION**
```
  (IL:LAMBDA (CONTEXT CHARCODE OPTIONS)                                          ; Edited  5-Jul-88 15:53 by woz
    (LET ((STRUCTURE (GET-SELECTED-STRUCTURE CONTEXT)))
        (COND
            (STRUCTURE (COND
                          ((FUNCALL *EDIT-FN* STRUCTURE OPTIONS)
                           (SET-SELECTION-NOWHERE (IL:FETCH (EDIT-CONTEXT SELECTION) IL:OF CONTEXT))
                           (SET-POINT-NOWHERE (IL:FETCH (EDIT-CONTEXT CARET-POINT) IL:OF CONTEXT)))))
            (T (FORMAT (GET-PROMPT-WINDOW CONTEXT)
                   "~%Select name of object to edit.")))
      T))
```

(**EVAL-SELECTION**
```
  (IL:LAMBDA (CONTEXT)                                                           ; Edited 29-Oct-87 15:14 by drc:
```

;;; evaluate the selected structure in the appropriate process, which should be stored in the EvalInProcess field of the context. If this field is NIL, then the
;;; process went away unexpectedly, so find an exec to eval in. This is dangerous: FIND.PROCESS 'EXEC IS NOT GUARANTEED!

```
    (LET* ((STRUCTURE (GET-SELECTED-STRUCTURE CONTEXT))
           (STRUCTURE-COPY (COPY-TREE STRUCTURE))
           (PROCESS (IL:FETCH EVAL-IN-PROCESS IL:OF CONTEXT))
           (PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
           (VALUE 'IL:NOBIND))
        (IL:TERPRI PROMPTWINDOW)
        (WHEN (NOT (IL:PROCESSP PROCESS))
            (IL:SETQ PROCESS (IL:REPLACE EVAL-IN-PROCESS IL:OF CONTEXT IL:WITH (IL:FIND.PROCESS 'IL:MOUSE))))
        (COND
```

```
            ((NULL STRUCTURE)
             (IL:|printout| PROMPTWINDOW T "Invalid selection for evaluation."))
            ((IL:LISTP STRUCTURE)
             (IL:SETQ VALUE (IL:RESETFORM (IL:TTY.PROCESS PROCESS)
                                  (IL:PROCESS.EVAL PROCESS '(IL:ERSETQ ,STRUCTURE)
                                           T)))
             (UNLESS (EQUAL STRUCTURE STRUCTURE-COPY)

                  ;; eval (DWIM) changed the structure

                  (REPLACE-NODE CONTEXT (IL:FETCH SELECT-NODE IL:OF (IL:FETCH SELECTION IL:OF CONTEXT))
                          (PARSE-NEW STRUCTURE CONTEXT)))
             (IF VALUE
                 (IL:SETQ VALUE (CAR VALUE))
                 (IL:SETQ VALUE 'IL:NOBIND)))
            ((IL:NUMBERP STRUCTURE)

             ;; make numbers eval to themselves, since PROCESS.EVALV won't work

             (IL:SETQ VALUE STRUCTURE))
            ((IL:ATOM STRUCTURE)
             (IL:SETQ VALUE (IL:PROCESS.EVALV PROCESS STRUCTURE))
             (WHEN (EQ VALUE 'IL:NOBIND)
                   (IL:|printout| PROMPTWINDOW T "Unbound atom: " IL:.P2 STRUCTURE)))
            (T (IL:SETQ VALUE STRUCTURE)))
        (WHEN (IL:NEQ VALUE 'IL:NOBIND)
              (COND
                 ((OR (IL:ATOM VALUE)
                      (IL:STRINGP VALUE))
                  (IL:|printout| PROMPTWINDOW T "Result: " IL:.P2 VALUE))
                 (T (SET-SELECTION-NOWHERE (IL:FETCH (EDIT-CONTEXT SELECTION) IL:OF CONTEXT))
                    (INSPECT VALUE)))))
    T))
```

(**EXPAND**
```
  (IL:LAMBDA (CONTEXT CHARCODE)                                    ; Edited  7-Jan-88 13:43 by DCB
```

;;; Replace the current selection with its macro-expansion, if any.

```
    (LET* ((PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
           (SELECTION (IL:FETCH SELECTION IL:OF CONTEXT))
           (POINT (IL:FETCH CARET-POINT IL:OF CONTEXT))
           (NODE (IL:FETCH SELECT-NODE IL:OF SELECTION)))
          (COND
             ((AND NODE (EQ (IL:FETCH SELECT-TYPE IL:OF SELECTION)
                            'STRUCTURE)
                        (NULL (IL:FETCH SELECT-START IL:OF SELECTION)))
              (LET ((STRUCTURE (IL:FETCH STRUCTURE IL:OF NODE))
                    EXPANSION)
                   (WHEN (CONSP STRUCTURE)

                         ;; we have a whole list structure node selected.  try to expand its definition

                         (IL:|printout| PROMPTWINDOW T "Looking for expansion...")
                         (IL:SETQ EXPANSION (IL:NLSETQ (IL:EDITGETD STRUCTURE)))
                         (COND
                            ((NULL EXPANSION)
                             (IL:|printout| PROMPTWINDOW T "Error during macro expansion."))
                            ((NOT (EQUAL (CAR EXPANSION)
                                         STRUCTURE))
                             (IL:TERPRI PROMPTWINDOW)
                             (REPLACE-NODE CONTEXT NODE (PARSE-NEW (CAR EXPANSION)
                                                          CONTEXT)))
                            (T (IL:|printout| PROMPTWINDOW T "No expansion found."))))))
             (T (IL:|printout| PROMPTWINDOW T "Can't expand this selection.")))))
    T))
```

(**EXTRACT-CURRENT-SELECTION**
```
  (IL:LAMBDA (CONTEXT)                                             ; Edited 27-Jun-88 15:30 by woz
    (CLOSE-OPEN-NODE CONTEXT)
    (LET* ((PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
           (SELECTION (IL:|fetch| SELECTION IL:|of| CONTEXT))
           (POINT (IL:|fetch| CARET-POINT IL:|of| CONTEXT))
           (NODE (IL:|fetch| SELECT-NODE IL:|of| SELECTION))
           SUBNODES SET-SELECTION?)
          (WHEN (AND (NULL NODE)
                     (IL:SETQ NODE (IL:|fetch| POINT-NODE IL:|of| POINT))
                     (EQ (IL:|fetch| POINT-TYPE IL:|of| POINT)
                         'STRUCTURE))

                ;; when you've only got a structure point extract from the list pointed within

                (SET-SELECTION-ME SELECTION CONTEXT NODE))
          (COND
             ((OR (NULL NODE)
                  (IL:NEQ (IL:|fetch| SELECT-TYPE IL:|of| SELECTION)
                          'STRUCTURE)
                  (IL:|fetch| SELECT-START IL:|of| SELECTION)
                  (IL:|fetch| SELECT-END IL:|of| SELECTION))
```

```
                            (IL:|printout| PROMPTWINDOW T "Select structure to extract."))
                        ((EQ 0 (CAR (IL:|fetch| SUB-NODES IL:|of| NODE)))              ; nothing to extract
                         (IL:|printout| PROMPTWINDOW T "Nothing to extract."))
                        ((EQ (IL:|fetch| NODE-TYPE IL:|of| NODE)
                             TYPE-COMMENT)
                         (LET ((START 0)
                               (STRING (THIRD (IL:FETCH STRUCTURE IL:OF NODE)))
                                STRUCTURE NEW-STRUCTURES)
                               (COND
                                  ((IL:NLSETQ (LOOP (IF (EQ :SEDIT-READ-END-FLG (MULTIPLE-VALUE-SETQ (STRUCTURE START)
                                                                                     (READ-FROM-STRING STRING NIL
                                                                                          :SEDIT-READ-END-FLG :START
                                                                                          START)))
                                                        (RETURN T)
                                                        (PUSH STRUCTURE NEW-STRUCTURES))))
                                   (SETQ SUBNODES (MAPCAR #'(LAMBDA (S)
                                                              (PARSE-NEW S CONTEXT))
                                                          (NREVERSE NEW-STRUCTURES)))
                                   (UNLESS (CDR SUBNODES)
                                       (SETQ SET-SELECTION? (CAR SUBNODES)))
                                   (PENDING-DELETE POINT SELECTION)
                                   (INSERT POINT CONTEXT SUBNODES))
                                  (T (FORMAT PROMPTWINDOW "~%Unreadable structure in comment.  Can't Extract.")))))
                        (T (IL:|replace| POINT-NODE IL:|of| POINT IL:|with| SELECTION)
                           (IL:|replace| POINT-TYPE IL:|of| POINT IL:|with| 'STRUCTURE)
                           (IL:SETQ SUBNODES (CDR (IL:|fetch| SUB-NODES IL:|of| NODE)))
                           (UNLESS (CDR SUBNODES)
                               (SETQ SET-SELECTION? (CAR SUBNODES)))
                           (RPLACD (IL:|fetch| SUB-NODES IL:|of| NODE)
                                   NIL)
                           (START-UNDO-BLOCK)
                           (UNDO-BY UNDO-EXTRACT NODE SUBNODES)                     ; replace with subnodes
                           (INSERT POINT CONTEXT (IL:COPY SUBNODES))
                           (END-UNDO-BLOCK)))
                  (WHEN SET-SELECTION?

                      ;; if only one subnode, leave it selected

                      (SET-SELECTION-ME SELECTION CONTEXT SET-SELECTION?)
                      (IL:|replace| PENDING-DELETE? IL:|of| SELECTION IL:|with| NIL)))

            ;; must return non-NIL if command executed

            T))
```

(**FIND-COMMENT**
```
  (IL:LAMBDA (NODE CONTEXT MIN-DEPTH LAST-SUBNODE)                           ; Edited  3-Dec-87 12:54 by DCB

;;; search starting with NODE for a node whose structure begins with a comment char .  move selection and point accordingly. return the node found,
;;; else NIL

      (WHEN NODE
          (IL:BIND (COMMENTCHAR IL:_ (IF (IL:LISTP IL:COMMENTFLG)
                                         (CAR IL:COMMENTFLG)
                                         IL:COMMENTFLG))
             IL:UNTIL (OR (NULL NODE)
                          (IL:ILESSP (IL:FETCH DEPTH IL:OF NODE)
                                 MIN-DEPTH)
                          (AND (EQ (IL:FETCH DEPTH IL:OF NODE)
                                   MIN-DEPTH)
                               (IL:IGREATERP (IL:FETCH SUB-NODE-INDEX IL:OF NODE)
                                      LAST-SUBNODE)))
             IL:DO (WHEN (EQ COMMENTCHAR (CAR (IL:FETCH STRUCTURE IL:OF NODE)))
                       (RETURN NODE))
                   (IL:SETQ NODE (NEXT-NODE NODE))))))
```

(**GET-MENU**
```
  (IL:LAMBDA (CONTEXT)                                                       ; Edited  7-Jul-87 09:28 by DCB
    (LET (MENU)
         (COND
            ((IL:SETQ MENU (IL:POP MENUS))
             (IL:FM.RESETMENU MENU))
            (T (IL:SETQ MENU (IL:FREEMENU MENU-DESCRIPTION "SEdit Command Menu"))
               (IL:WINDOWADDPROP MENU 'IL:CLOSEFN 'MENU-CLOSEFN)
               (IL:WINDOWPROP MENU 'IL:FM.DONTRESHAPE T)))
         (MENU-INIT-STATE MENU CONTEXT)
         MENU)))
```

(**EDIT-HELP**
```
  (IL:LAMBDA (CONTEXT)                                                       ; Edited  7-Jul-87 09:29 by DCB
    (CLOSE-OPEN-NODE CONTEXT)
    (LET* ((POINT (IL:FETCH CARET-POINT IL:OF CONTEXT))
           (NODE (IL:FETCH POINT-NODE IL:OF POINT)))
          (WHEN (AND (IL:TYPE? EDIT-NODE NODE)
                     (IL:LITATOM (IL:FETCH STRUCTURE IL:OF NODE))
                     (EQ (IL:FETCH POINT-INDEX POINT)
```

```
                       (IL:NCHARS (IL:FETCH STRUCTURE IL:OF NODE))))
                                                                    ; if at end of this node, change to structure point.
              (INSERT POINT CONTEXT NIL)))
     (LET* ((FNAME (SELECTED-FN-NAME CONTEXT))
            (PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
            ARGS)
           (IF FNAME
               (IF (IL:SETQ ARGS (IL:NLSETQ (IL:SMARTARGLIST FNAME T)))
                   (COND
                     ((IL:ILEQ (IL:STRINGWIDTH (IL:SETQ ARGS (CONS FNAME (CAR ARGS))))
                               (IL:WINDOWPROP PROMPTWINDOW 'IL:WIDTH))
                                                                    ; will fit in attached window
                      (IL:|printout| PROMPTWINDOW T ARGS))
                     (T                                             ; put in main promptwindow
                        (IL:TERPRI PROMPTWINDOW)
                        (IL:|printout| IL:PROMPTWINDOW T ARGS)))
                   (IL:|printout| PROMPTWINDOW T "Arguments not available for " FNAME))
               (IL:|printout| PROMPTWINDOW T "Select function you want the arguments for.")))
        T))
```

(**HELPMENU**
```
  (IL:LAMBDA (CONTEXT)                                             ; Edited 24-May-88 14:20 by woz
     (LET ((MENU (IL:FETCH HELP-MENU IL:OF (IL:FETCH ENVIRONMENT IL:OF CONTEXT)))
           (PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
           COMMAND)
          (WHEN (LISTP MENU)
              (FORMAT PROMPTWINDOW "~%Creating menu, please wait...")

          ;; build the popup menu info.  the lists of menu-items, menu-left strings, and menu-right strings are in the MENU list.  take it apart, then
          ;; build the menu.  this information was compiled in create-command-table, but the menu gets built when first used (so the font ends up
          ;; right if the user changed it).

              (LET* ((FONT (IL:FONTCREATE IL:MENUFONT))
                     (MENU-ITEMS (FIRST MENU))
                     (EQUALIZED-MENU-LEFT (EQUALIZE-STRING-WIDTHS (SECOND MENU)
                                                                  FONT))
                     (MENU-RIGHT (THIRD MENU))
                     ITEMWIDTH ITEMS)

          ;; figure out the width of the left column, including the tab, then set the menu width.  Do this by finding the first tab stop after the
          ;; shortest stringwidth in EQUALIZED-MENU-LEFT.  We know that the widths of each equalized string are within one space
          ;; width of each other, and since a tab is bigger than a space, we know that this tab stop is the first after all of the strings,
          ;; allowing tabulation.

          ;; There is a strange feature of the menu code that starts printing lables at 1 instead of zero, which changes the relative tab
          ;; stop position.  This shift can cause a tab stop to fall in between the shortest and longest equalized strings.  So we have to
          ;; see if our chosen tab stop is within one pixel of the longest string, and if so, pad the strings with an extra space to jump them
          ;; all past that tab stop.

                 (DO* ((LEFT-WIDTH (MINIMUM-STRING-WIDTH EQUALIZED-MENU-LEFT FONT))
                       (TAB-WIDTH (IL:STRINGWIDTH "        " FONT))
                       (TAB-COLUMN TAB-WIDTH (+ TAB-COLUMN TAB-WIDTH)))
                      ((> TAB-COLUMN LEFT-WIDTH)

                       ;; check for the stupid menu case:

                       (WHEN (= (1- TAB-COLUMN)
                                (MAXIMUM-STRING-WIDTH EQUALIZED-MENU-LEFT FONT))
                           (SETQ EQUALIZED-MENU-LEFT (EQUALIZE-STRING-WIDTHS EQUALIZED-MENU-LEFT FONT NIL
                                                                            TAB-COLUMN))
                           (INCF TAB-COLUMN TAB-WIDTH))
                       (SETQ ITEMWIDTH (+ TAB-COLUMN (MAXIMUM-STRING-WIDTH MENU-RIGHT FONT))))
                    NIL)
                 ;; construct the menu strings and the menu items.

                 (DO ((LEFT EQUALIZED-MENU-LEFT (REST LEFT))
                      (RIGHT MENU-RIGHT (REST RIGHT))
                      (ITEM MENU-ITEMS (REST ITEM)))
                     ((NULL ITEM)
                      (SETQ ITEMS (NREVERSE MENU-ITEMS)))
                    (PUSH (CONCATENATE 'STRING (FIRST LEFT)
                                  (STRING #\Tab)
                                  (FIRST RIGHT))
                          (FIRST ITEM)))
                 (IL:REPLACE HELP-MENU IL:OF (IL:FETCH ENVIRONMENT IL:OF CONTEXT)
                        IL:WITH (SETQ MENU (IL:CREATE IL:MENU
                                                  IL:ITEMS IL:_ ITEMS
                                                  IL:ITEMWIDTH IL:_ ITEMWIDTH
                                                  IL:CHANGEOFFSETFLG IL:_ 'IL:Y
                                                  IL:MENUOFFSET IL:_ (CONS -1 0)
                                                  IL:TITLE IL:_ "Commands")))))
              (WHEN (SETQ COMMAND (IL:MENU MENU))
                  (TERPRI PROMPTWINDOW)
                  (AWAKE-COMMAND-PROCESS CONTEXT COMMAND)))))
```

(**INPUT-DOT**
```
  (IL:LAMBDA (CONTEXT CHARCODE)                                    ; Edited  7-Jul-87 09:29 by DCB
```

;;; handle input of a dot. cases:

;;; (1) structure selection; might be a quoted gap to be ugraded, otherwise just a node to delete in a list to be dotted.

;;; (2) structure point; in a list to be dotted.

;;; (3) atom point; might be at the beginning of a quote to be ugraded, otherwise just insert the dot.

```
        (LET* ((POINT (IL:FETCH CARET-POINT IL:OF CONTEXT))
               (NODE (IL:FETCH POINT-NODE IL:OF POINT)))
            (COND
               ((IL:TYPE? EDIT-SELECTION NODE)
                (LET ((SELECTION NODE))

                     ;; if we're at a structure selection, this is interesting.  otherwise, let the char handler input the dot.

                     (WHEN (EQ 'STRUCTURE (IL:FETCH SELECT-TYPE IL:OF SELECTION))
                         (COND
                            ((EQ TYPE-QUOTE (IL:FETCH NODE-TYPE IL:OF (IF (IL:FETCH SELECT-START IL:OF SELECTION)
                                                                          (IL:FETCH SELECT-NODE IL:OF SELECTION)
                                                                          (IL:FETCH SUPER-NODE
                                                                             IL:OF (IL:FETCH SELECT-NODE
                                                                                      IL:OF SELECTION)))))

                             ;; we're in a quote form.  let the quote handler check for a comma-dot

                             (INPUT-QUOTE CONTEXT CHARCODE 'COMMA-DOT)
                             T)
                            (T
                               ;; just at a pending delete selection.  delete it and try to dot the list.

                               (DELETE-NODES (IL:FETCH SELECT-NODE IL:OF SELECTION)
                                       CONTEXT
                                       (IL:FETCH SELECT-START IL:OF SELECTION)
                                       (IL:FETCH SELECT-END IL:OF SELECTION)
                                       POINT)
                               (DOT-THIS-LIST CONTEXT)
                               T)))))
               ((AND NODE (EQ 'STRUCTURE (IL:FETCH POINT-TYPE IL:OF POINT)))
                ;; normal case of dot input at a structure point in a list

                (DOT-THIS-LIST CONTEXT)
                T)
               ((AND NODE (EQ 'ATOM (IL:FETCH POINT-TYPE IL:OF POINT))
                     (EQ 0 (IL:FETCH POINT-INDEX IL:OF POINT)))
                ;; at the beginning of an atom.  check if it's a comma quote, otherwise, just input

                (LET ((SUPER-NODE (IL:FETCH SUPER-NODE IL:OF NODE)))
                     (WHEN (AND (EQ TYPE-QUOTE (IL:FETCH NODE-TYPE IL:OF SUPER-NODE))
                                (EQ (QUOTE-WRAPPER 'IL:COMMA)
                                    (CAR (IL:FETCH STRUCTURE IL:OF SUPER-NODE))))
                         ;; we're at the beginning of a COMMA quote atom that wants to be upgraded

                         (CHANGE-QUOTE SUPER-NODE CONTEXT 'COMMA-DOT)
                         T)))))))
```

(**INPUT-ESCAPE**
```
    (IL:LAMBDA (CONTEXT)                                                          ; Edited 17-Nov-87 13:35 by DCB
```

;;; dynamically set this.char.escaped true, so that next time through the loop, it knows it's getting an escaped char

```
    (IL:SETQ THIS-CHAR-ESCAPED T)))
```

(**INPUT-NORMAL-CHAR**
```
    (IL:LAMBDA (CONTEXT CHAR)                                                     ; Edited  7-Jul-87 09:29 by DCB
        (COND
           ((AND (IL:IGREATERP CHAR 255)
                 (IL:ILESSP CHAR 512))
            ;; this is a meta-character that wasn't recognized as a command.  don't insert it!

            (IL:|printout| (GET-PROMPT-WINDOW CONTEXT)
                   T "Unknown command: Meta-" (IL:CHARACTER (IL:IDIFFERENCE CHAR 256))))
           (T (LET ((POINT (IL:FETCH (EDIT-CONTEXT CARET-POINT) IL:OF CONTEXT))
                    (POINT-TYPE (TYPE-OF-INPUT CONTEXT)))
                  (IL:SETQ CHAR (IL:CHARACTER CHAR))
                  (WHEN (IL:NEQ POINT-TYPE 'STRING)
                      (COND
                         (THIS-CHAR-ESCAPED

                            ;; prepend an escape character                    ; read table specific
                            (IL:SETQ CHAR (IL:CONCAT (IL:CHARACTER (ESCAPE-CHAR))
                                                  CHAR)))
                         ((AND (IL:FETCH (READTABLEP IL:CASEINSENSITIVE) IL:OF *READTABLE*)
                               (IL:NEQ POINT-TYPE 'ESC-ATOM))
                          (IL:SETQ CHAR (IF (OR (EQ POINT-TYPE 'STRUCTURE)
                                                (EQ *PRINT-CASE* ':UPCASE))
                                            (IL:U-CASE CHAR)
                                            (IL:L-CASE CHAR))))))
                  (IL:SELECTQ POINT-TYPE
                      (STRUCTURE
```

```
                              ;; first mark that we're starting an atom, because the reparser needs to know when inserting in a lambda
                              ;; arglist slot whether or not to reparse it as a list.  THIS IS UGLY, but it works.

                              (IL:REPLACE ATOM-STARTED IL:OF CONTEXT IL:WITH T)
                              (INSERT POINT CONTEXT CHAR)
                              (IL:REPLACE ATOM-STARTED IL:OF CONTEXT IL:WITH (IL:FETCH POINT-NODE IL:OF POINT))
                              (IL:REPLACE ATOM-STARTED-UNDO-POINTER IL:OF CONTEXT IL:WITH (IL:FETCH UNDO-LIST
                                                                                                IL:OF CONTEXT)))
                 ((ATOM ESC-ATOM)
                      (LET ((NODE (IL:FETCH POINT-NODE IL:OF POINT))
                             IL:WHERE)
                          (COND
                              ((IL:TYPE? EDIT-NODE NODE)
                               (IL:SETQ IL:WHERE POINT))
                              (T  ;; the pending-delete case.  the PointNode actually points to a selection framing the material to be
                                  ;; replaced

                                  (IL:SETQ NODE (IL:FETCH SELECT-NODE IL:OF (IL:SETQ IL:WHERE NODE)))))
                          (INSERT-STRING NODE CONTEXT IL:WHERE CHAR POINT)
                          (SET-SELECTION-NOWHERE (IL:FETCH SELECTION IL:OF CONTEXT))))
                 (STRING (INSERT POINT CONTEXT CHAR))
                 (NIL)
                 (IL:SHOULDNT "bad point type")))))
         (SET-SELECTION-NOWHERE (IL:FETCH (EDIT-CONTEXT SELECTION) IL:OF CONTEXT)))))
```

(**INPUT-QUOTE**
```
  (IL:LAMBDA (CONTEXT CHARCODE QUOTE-TYPE)                                    ; Edited 19-Nov-87 15:28 by DCB
    (IL:SELECTQ (TYPE-OF-INPUT CONTEXT)
         (STRUCTURE (CLOSE-OPEN-NODE CONTEXT)
                  (COND
                      ((IL:FMEMB QUOTE-TYPE '(COMMA-AT COMMA-DOT))

                       ;; check if we're in a COMMA quote to be upgraded

                       (LET* ((SELECTION (IL:FETCH SELECTION IL:OF CONTEXT))
                               (NODE (IL:FETCH SELECT-NODE IL:OF SELECTION))
                               (SUPER-NODE))
                            (WHEN (AND NODE (IL:SETQ SUPER-NODE (IL:FETCH SUPER-NODE IL:OF NODE))
                                       (EQ TYPE-GAP (IL:FETCH NODE-TYPE IL:OF NODE))
                                       (EQ TYPE-QUOTE (IL:FETCH NODE-TYPE IL:OF SUPER-NODE))
                                       (EQ (QUOTE-WRAPPER 'IL:COMMA)
                                           (CAR (IL:FETCH STRUCTURE IL:OF SUPER-NODE))))

                                 ;; we're in the middle of typing in a COMMA quote form that wants to be upgraded

                                 (CHANGE-QUOTE SUPER-NODE CONTEXT QUOTE-TYPE)
                                 T)))
                      (T (INSERT-QUOTED-GAP CONTEXT CHARCODE QUOTE-TYPE)
                         T)))
         (ATOM ;; check if we're at the beginning of an atom to quote.  otherwise, let the quote be inserted normally

                  (LET* ((POINT (IL:FETCH CARET-POINT IL:OF CONTEXT))
                          (NODE (IL:FETCH POINT-NODE IL:OF POINT))
                          (SUPER-NODE (AND (IL:TYPE? EDIT-NODE NODE)
                                           (IL:FETCH SUPER-NODE IL:OF NODE))))
                       (COND
                          ((AND SUPER-NODE (EQ 0 (IL:FETCH POINT-INDEX POINT)))
                           (COND
                              ((EQ QUOTE-TYPE 'COMMA-AT)

                               ;; this is tricky.  we got an @ at the beginning of an atom. if it's in a COMMA quote, then upgrade, otherwise
                               ;; insert the @ as part of the atom.

                               (WHEN (AND (EQ TYPE-QUOTE (IL:FETCH NODE-TYPE IL:OF SUPER-NODE))
                                          (EQ (QUOTE-WRAPPER 'IL:COMMA)
                                              (CAR (IL:FETCH STRUCTURE IL:OF SUPER-NODE))))
                                 (CHANGE-QUOTE SUPER-NODE CONTEXT 'COMMA-AT)
                                 T))
                              (T (SET-SELECTION-ME (IL:FETCH SELECTION IL:OF CONTEXT)
                                        CONTEXT NODE)
                                 (QUOTE-CURRENT-SELECTION CONTEXT CHARCODE QUOTE-TYPE)
                                 (SET-SELECTION-NOWHERE (IL:FETCH SELECTION IL:OF CONTEXT))
                                 (SET-POINT POINT CONTEXT NODE)
                                 T)))
                          ((AND SUPER-NODE (EQ QUOTE-TYPE 'QUOTE)
                                (EQ 1 (IL:FETCH POINT-INDEX POINT))
                                (EQ (IL:CHARCODE \#)
                                    (IL:CHCON1 (IL:FETCH POINT-STRING IL:OF POINT))))
                           ;; this is tricky.  We are adding the ' part of #', so we want to function wrap the rest of this string (or gap it if it's
                           ;; empty).

                           (COND
                              ((EQ 1 (IL:NCHARS (IL:FETCH POINT-STRING IL:OF POINT)))
                               ;; close the node, get rid of it, and replace it with a quoted gap.  Oh yeah, do this undoably by just closing and
                               ;; calling an undoable thing,

                               (CLOSE-OPEN-NODE CONTEXT)
                               (SET-SELECTION-ME (IL:FETCH SELECTION IL:OF CONTEXT)
                                        CONTEXT NODE)
                               (PENDING-DELETE POINT (IL:FETCH SELECTION IL:OF CONTEXT))
```

```
                                   (INSERT-QUOTED-GAP CONTEXT NIL 'FUNCTION)
                                T)
                              (T
                                 ;; remove the #, close the node, wrap it with function, and put point at the first character.  Oh yeah, do this
                                 ;; undoably.

                                 (START-UNDO-BLOCK)
                                 (REPLACE-STRING NODE CONTEXT 1 1 "" POINT (IL:FETCH POINT-STRING IL:OF POINT)
                                        'ATOM)
                                 (SET-SELECTION-ME (IL:FETCH SELECTION IL:OF CONTEXT)
                                        CONTEXT NODE)
                                 (QUOTE-CURRENT-SELECTION CONTEXT NIL 'FUNCTION)
                                 (SET-SELECTION-NOWHERE (IL:FETCH SELECTION IL:OF CONTEXT))
                                 (SET-POINT POINT CONTEXT NODE NIL NIL NIL 'ATOM)
                                 (END-UNDO-BLOCK)
                                 T))))))
            NIL)))
```

(**INPUT-SQUARE-BRACKET**
```
   (IL:LAMBDA (CONTEXT CHARCODE)                                     ; Edited  7-Jul-87 09:29 by DCB
      (WHEN (IL:NEQ (TYPE-OF-INPUT CONTEXT)
                    'STRING)
         (LET ((PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT)))
              (IL:|printout| PROMPTWINDOW T "SEdit can't handle square brackets.  Ignoring rest of input.")
              (IL:FLASHWINDOW PROMPTWINDOW)
              (IL:CLEARBUF T)
              T))))
```

(**INPUT-STRINGDELIM**
```
   (IL:LAMBDA (CONTEXT)                                              ; Edited 17-Nov-87 13:35 by DCB
      (COND
         ((EQ (TYPE-OF-INPUT CONTEXT)
              'STRING)

          ;; split or close this string

          (LET* ((POINT (IL:FETCH CARET-POINT IL:OF CONTEXT))
                 (NODE (IL:FETCH POINT-NODE IL:OF POINT)))
                (WHEN (IL:TYPE? EDIT-SELECTION NODE)
                   (IL:SETQ NODE (IL:FETCH SELECT-NODE IL:OF NODE)))
                (WHEN (EQ (IL:FETCH NODE-TYPE IL:OF NODE)
                          TYPE-STRING)
                   (INSERT POINT CONTEXT NIL)
                   (SET-SELECTION-NOWHERE (IL:FETCH (EDIT-CONTEXT SELECTION) IL:OF CONTEXT))
                   T)))
         (T ;; insert a new string

          (LET ((NEW-STRING (IL:ALLOCSTRING 0))
                (POINT (IL:FETCH (EDIT-CONTEXT CARET-POINT) IL:OF CONTEXT)))
               (IL:SETQ NEW-STRING (CREATE-SIMPLE-NODE NEW-STRING (IL:FETCH ENVIRONMENT IL:OF CONTEXT)
                                            TYPE-STRING NEW-STRING T (IL:FETCH DEFAULT-FONT
                                                                          IL:OF (IL:FETCH ENVIRONMENT IL:OF CONTEXT)))
                    )
               (INSERT POINT CONTEXT NEW-STRING)
               (WHEN (NOT (DEAD-NODE? NEW-STRING))
                  (IL:REPLACE POINT-NODE IL:OF POINT IL:WITH NEW-STRING)
                  (IL:REPLACE POINT-INDEX IL:OF POINT IL:WITH 0)
                  (IL:REPLACE POINT-TYPE IL:OF POINT IL:WITH 'STRING)
                  (IL:REPLACE POINT-STRING IL:OF POINT IL:WITH (IL:FETCH STRUCTURE IL:OF NEW-STRING))
                  (SET-SELECTION-NOWHERE (IL:FETCH (EDIT-CONTEXT SELECTION) IL:OF CONTEXT))
                  (IL:REPLACE ATOM-STARTED IL:OF CONTEXT IL:WITH NEW-STRING)
                  (IL:REPLACE ATOM-STARTED-UNDO-POINTER IL:OF CONTEXT IL:WITH (IL:FETCH UNDO-LIST IL:OF CONTEXT))
                  ))
          T))))
```

(**INPUT-TOKENDELIM**
```
   (IL:LAMBDA (CONTEXT CHARCODE)                                     ; Edited  7-Jul-87 09:29 by DCB
      (LET ((POINT (IL:FETCH CARET-POINT IL:OF CONTEXT)))
           (IL:SELECTQ (TYPE-OF-INPUT CONTEXT)
                 (ATOM (INSERT POINT CONTEXT NIL)
                       (SET-SELECTION-NOWHERE (IL:FETCH (EDIT-CONTEXT SELECTION) IL:OF CONTEXT)))
                 (STRUCTURE (WHEN (NOT (IL:FETCH PENDING-DELETE? IL:OF (IL:FETCH SELECTION IL:OF CONTEXT)))

                                 ;; this test so that delims don't do anything on pending delete gaps in particular, to avoid <dot> <space> or
                                 ;; <quote> <space> wasting the gap.  i don't think it will hurt the other cases.

                                 (INSERT POINT CONTEXT NIL)
                                 (SET-SELECTION-NOWHERE (IL:FETCH (EDIT-CONTEXT SELECTION) IL:OF CONTEXT))))
                 ((STRING ESC-ATOM)
                       (IF (AND (EQ CHARCODE (IL:CHARCODE IL:CR))
                                (EQ-POINT-TYPE POINT TYPE-COMMENT))
                           (INSERT POINT CONTEXT NIL)
                           (INSERT POINT CONTEXT (IL:CHARACTER CHARCODE)))
                       (SET-SELECTION-NOWHERE (IL:FETCH (EDIT-CONTEXT SELECTION) IL:OF CONTEXT)))
                 (NIL)
                 (IL:SHOULDNT "bad point type")))
      T))
```

(INSERT-MULTI-ESCAPE
```
  (IL:LAMBDA (CONTEXT CHAR)                                              ; Edited  7-Jul-87 09:29 by DCB
    (LET ((POINT (IL:FETCH CARET-POINT IL:OF CONTEXT))
          (TYPE (TYPE-OF-INPUT CONTEXT))
           NODE IL:WHERE)
         (COND
            ((EQ TYPE 'STRUCTURE)
             (INSERT POINT CONTEXT (IL:ALLOCSTRING 2 CHAR))
             (IL:REPLACE POINT-INDEX IL:OF POINT IL:WITH 1)
             (IL:REPLACE POINT-TYPE IL:OF POINT IL:WITH 'ESC-ATOM))
            ((OR (EQ TYPE 'ATOM)
                 (EQ TYPE 'ESC-ATOM))
             (IF (IL:TYPE? EDIT-NODE (IL:SETQ NODE (IL:FETCH POINT-NODE IL:OF POINT)))
                 (IF (AND (EQ TYPE 'ESC-ATOM)
                          (EQ (IL:NTHCHARCODE (IL:FETCH POINT-STRING IL:OF POINT)
                                   (IL:ADD1 (IL:FETCH POINT-INDEX IL:OF POINT)))
                              CHAR))
                     (IL:ADD (IL:FETCH POINT-INDEX IL:OF POINT)
                             1)
                     (IL:SETQ IL:WHERE POINT))
                 (IL:SETQ NODE (IL:FETCH SELECT-NODE IL:OF (IL:SETQ IL:WHERE NODE))))
             (WHEN IL:WHERE
                 (INSERT-STRING NODE CONTEXT IL:WHERE (IL:ALLOCSTRING 2 CHAR)
                        POINT)
                 (IL:ADD (IL:FETCH POINT-INDEX IL:OF POINT)
                         -1))
             (IL:REPLACE POINT-TYPE IL:OF POINT IL:WITH (IF (EQ (IL:FETCH POINT-TYPE IL:OF POINT)
                                                                'ATOM)
                                                            'ESC-ATOM
                                                            'ATOM))
             (SET-SELECTION-NOWHERE (IL:FETCH SELECTION IL:OF CONTEXT))
             T)))))
```

(INSERT-SPECIAL-CHARACTER
```
  (IL:LAMBDA (CONTEXT CHAR)                                              ; Edited  7-Jul-87 09:29 by DCB
```

;;; insert a special character (e.g. the package delimiter) without escaping it

```
    (LET ((POINT (IL:FETCH CARET-POINT IL:OF CONTEXT))
          (STRING (IL:ALLOCSTRING 1 CHAR)))
         (IL:SELECTQ (TYPE-OF-INPUT CONTEXT)
             (ATOM (LET ((NODE (IL:FETCH POINT-NODE IL:OF POINT))
                          IL:WHERE)
                        (COND
                            ((IL:TYPE? EDIT-NODE NODE)
                             (IL:SETQ IL:WHERE POINT))
                            (T  ;; the pending-delete case.  the PointNode actually points to a selection framing the material to be replaced

                                (IL:SETQ NODE (IL:FETCH SELECT-NODE IL:OF (IL:SETQ IL:WHERE NODE)))))
                        (INSERT-STRING NODE CONTEXT IL:WHERE STRING POINT)
                        (SET-SELECTION-NOWHERE (IL:FETCH SELECTION IL:OF CONTEXT))
                        T))
             (STRUCTURE  ;; LET ((new.node (fetch PointNode of point))) (replace AtomStarted of context with new.node) (replace
                         ;; AtomStartedUndoPointer of context with (fetch UndoList of context)) (open.litatom context new.node string)
                         ;; (replace OpenNodeChanged? of context with T) (adjust.width new.node context (STRINGWIDTH string) (fetch
                         ;; Font of (CAR (fetch LinearForm of new.node)))) (replace PointIndex of point with 1) (replace PointString of point
                         ;; with string) T

                         (INSERT POINT CONTEXT STRING)
                         T)
             NIL))))
```

(INSPECT-SELECTION
```
  (IL:LAMBDA (CONTEXT)                                                   ; Edited 17-Nov-87 13:36 by DCB
    (LET ((STRUCTURE (GET-SELECTED-STRUCTURE CONTEXT)))
         (COND
            (STRUCTURE (SET-SELECTION-NOWHERE (IL:FETCH (EDIT-CONTEXT SELECTION) IL:OF CONTEXT))
                    (SET-POINT-NOWHERE (IL:FETCH (EDIT-CONTEXT CARET-POINT) IL:OF CONTEXT))

                    ;; update context

                    (WHEN (NULL (IL:NLSETQ (INSPECT STRUCTURE)))
                        (IL:|printout| (GET-PROMPT-WINDOW CONTEXT)
                               T "Inspection aborted.")))
            (T (IL:|printout| (GET-PROMPT-WINDOW CONTEXT)
                      T "Select object to inspect."))))
    T))
```

(JOIN
```
  (IL:LAMBDA (CONTEXT)                                                   ; Edited  7-Jul-87 09:36 by DCB
    (LET* ((PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
           (SELECTION (IL:FETCH SELECTION IL:OF CONTEXT))
           (POINT (IL:FETCH CARET-POINT IL:OF CONTEXT))
```

```
                  (NODE (IL:FETCH SELECT-NODE IL:OF SELECTION))
                  (START (IL:FETCH SELECT-START IL:OF SELECTION))
                  (END (IL:FETCH SELECT-END IL:OF SELECTION))
                  (COMMENT-LEVEL 1)
                 SUBNODES TYPE NEW-STRUCTURE NEW-NODE)
              (CLOSE-OPEN-NODE CONTEXT)
              (COND
                  ((NOT (AND NODE START END (IL:NEQ START END)
                             (EQ (IL:FETCH SELECT-TYPE IL:OF SELECTION)
                                 'STRUCTURE)))
                   (IL:|printout| PROMPTWINDOW T "Select items to join."))
                  ((AND (IL:SETQ TYPE (IL:FETCH NAME IL:OF (IL:FETCH NODE-TYPE IL:OF (SUBNODE START NODE))))
                        (IL:FMEMB TYPE (IL:CONSTANT '(QUOTE UNKNOWN GAP ROOT DOTLIST))))
                   (IL:|printout| PROMPTWINDOW T "Can't join things of this type."))
                  (T (IL:SETQ SUBNODES (IL:FETCH SUB-NODES IL:OF NODE))
                     (PENDING-DELETE POINT SELECTION)
                     (START-UNDO-BLOCK)
                     (IL:SELECTQ TYPE
                         ((LITATOM STRING)  ;; for these types, each node must be of the same SEdit type

                              (IL:SETQ NEW-STRUCTURE
                               (IL:FOR INDEX IL:FROM START IL:TO END IL:AS SUBNODE IL:IN (IL:NTH (CDR SUBNODES)
                                                                                             START)
                                   IL:COLLECT (WHEN (NOT (IL:FMEMB (IL:FETCH NAME IL:OF (IL:FETCH NODE-TYPE
                                                                                          IL:OF SUBNODE))
                                                             (IL:CONSTANT '(LITATOM STRING))))
                                                (IL:|printout| PROMPTWINDOW T "Each item to join must be of the
                                                      same type.")
                                                (RETURN))
                                          (IL:FETCH STRUCTURE IL:OF SUBNODE)))
                              (WHEN NEW-STRUCTURE
                                  (COND
                                      ((IL:NUMBERP (CAR NEW-STRUCTURE))
                                       (IL:|printout| PROMPTWINDOW T "Can't join numbers."))
                                      (T (IL:SETQ NEW-NODE (PARSE-NEW (IF (EQ TYPE 'LITATOM)
                                                                          (INTERN (IL:CONCATLIST NEW-STRUCTURE)
                                                                                  (SYMBOL-PACKAGE (CAR NEW-STRUCTURE
                                                                                                  )))
                                                                          (IL:CONCATLIST NEW-STRUCTURE))
                                                                      CONTEXT))
                                         (INSERT POINT CONTEXT NEW-NODE)))))
                         (COMMENT  ;; for comments, each node must be of the same SEdit type

                              (IL:SETQ NEW-STRUCTURE (IL:FOR INDEX IL:FROM START IL:TO END IL:AS SUBNODE
                                                          IL:IN (IL:NTH (CDR SUBNODES)
                                                                      START)
                                                          IL:JOIN (WHEN (IL:NEQ (IL:FETCH NAME
                                                                                     IL:OF (IL:FETCH NODE-TYPE
                                                                                               IL:OF SUBNODE))
                                                                              'COMMENT)
                                                                      (IL:|printout| PROMPTWINDOW T "Each item to
                                                                            join must be of the same type.")
                                                                      (RETURN))
                                                              (IL:SETQ COMMENT-LEVEL
                                                               (IL:IMAX COMMENT-LEVEL (IL:FETCH UNASSIGNED
                                                                                           IL:OF SUBNODE)))
                                                              (COND
                                                                  ((EQ INDEX END)
                                                                   (CDDR (IL:FETCH STRUCTURE IL:OF SUBNODE)))
                                                                  (T  ;; add space between comments

                                                                      (LIST (CADDR (IL:FETCH STRUCTURE
                                                                                      IL:OF SUBNODE))
                                                                            "  ")))))
                              (WHEN NEW-STRUCTURE
                                  (IL:SETQ NEW-STRUCTURE (LIST 'IL:* (CAR (IL:NTH COMMENT-MARKERS COMMENT-LEVEL
                                                                              ))
                                                               (IL:APPLY* 'IL:CONCATLIST NEW-STRUCTURE)))
                              (IL:SETQ NEW-NODE (PARSE-NEW NEW-STRUCTURE CONTEXT))
                              (INSERT POINT CONTEXT NEW-NODE)))
                         (PROGN  ;; for the rest, the structures must all be listp's

                              (COND
                                  ((IL:FOR INDEX IL:FROM START IL:TO END IL:AS SUBNODE
                                        IL:IN (IL:NTH (CDR SUBNODES)
                                                    START)
                                        IL:THEREIS (NOT (IL:LISTP (IL:FETCH STRUCTURE IL:OF SUBNODE))))
                                   (IL:|printout| PROMPTWINDOW T "Each item to join must be of the same type."))
                                  (T (IL:SETQ NEW-NODE (SUBNODE START NODE))
                                     (SET-POINT POINT CONTEXT NEW-NODE (CAR (IL:FETCH SUB-NODES IL:OF NEW-NODE))
                                          T
                                          (CAR (LAST (IL:FETCH SUB-NODES IL:OF NEW-NODE)))
                                          'STRUCTURE)
                                     (IL:FOR INDEX IL:FROM (IL:ADD1 START) IL:TO END IL:AS SUBNODE
                                         IL:IN (IL:NTH (CDR SUBNODES)
                                                     (IL:ADD1 START))
                                         IL:DO (IL:SETQ NEW-STRUCTURE (CDR (IL:FETCH SUB-NODES IL:OF SUBNODE)))
```

```
                                        (DELETE-NODES SUBNODE CONTEXT 1 (CAR (IL:FETCH SUB-NODES IL:OF SUBNODE
                                                                                                     )))
                                        (INSERT POINT CONTEXT NEW-STRUCTURE))
                                (DELETE-NODES NODE CONTEXT (IL:ADD1 START)
                                        END)))))
                (WHEN NEW-NODE
                    (SET-SELECTION-ME SELECTION CONTEXT NEW-NODE)
                    (IL:REPLACE PENDING-DELETE? IL:OF SELECTION IL:WITH NIL)
                    (SET-POINT POINT CONTEXT NEW-NODE NIL T NIL 'STRUCTURE))
                (END-UNDO-BLOCK))))
        T))
```

(**MENU-CLOSEFN**
```
  (IL:LAMBDA (W)                                                              ; Edited  7-Jul-87 09:36 by DCB
```

;;; must be called before menu is detached from sedit.

```
      (IL:PUSH MENUS W)
      (IL:WINDOWPROP (IL:MAINWINDOW W)
          'MENU NIL)))
```

(**MENU-FIND-SELECTEDFN**
```
  (IL:LAMBDA (ITEM WINDOW BUTTONS)                                            ; Edited 17-Jul-87 10:12 by DCB
    (LET ((FIND-ITEM (IL:LISTGET (IL:FM.ITEMPROP ITEM 'IL:LINKS)
                            'IL:EDIT))
          (CONTEXT (IL:WINDOWPROP (IL:MAINWINDOW WINDOW)
                        'EDIT-CONTEXT)))
        (COND
          ((OR (IL:EQUAL (IL:FM.ITEMPROP FIND-ITEM 'IL:LABEL)
                        "")
               (EQ (CAR BUTTONS)
                   'IL:RIGHT))
```
             ;; need new stuff to find
```
           (IL:FM.EDITITEM FIND-ITEM WINDOW T))
```
          (T   ;; call find with an extra argument of the stuff to find
```
              (MENU-SELECTEDFN ITEM WINDOW BUTTONS 'FIND (LIST (IL:FM.ITEMPROP FIND-ITEM 'IL:LABEL)))
              (IL:TTY.PROCESS (IL:WINDOWPROP (IL:MAINWINDOW WINDOW)
                                    'IL:PROCESS)))))))
```

(**MENU-INIT-STATE**
```
  (IL:LAMBDA (MENU CONTEXT)                                                   ; Edited  7-Jul-87 09:38 by DCB
```

;;; initialize menu profile entries. will be called by either under command loop, or under building new window, either case under sedit's profile, so
;;; references to *print* variables are okay.

```
      (LET* ((PACKAGE-NAME (PACKAGE-NAME *PACKAGE*))
             (PRINT-BASE *PRINT-BASE*)
             (*PRINT-BASE* 10))
```
            ;; want to display *PRINT-BASE* in print base 10, so must cache and rebind it.
```
            (IL:FM.CHANGESTATE 'PRINTBASE-VALUE-ITEM PRINT-BASE MENU)
            (IL:FM.ITEMPROP (IL:FM.GETITEM 'PRINTBASE-ITEM NIL MENU)
                    'PRINTBASE PRINT-BASE)
            (IL:FM.CHANGELABEL 'PACKAGE-NAME-ITEM PACKAGE-NAME MENU)
            (IL:FM.ITEMPROP (IL:FM.GETITEM 'PACKAGE-ITEM NIL MENU)
                    'PACKAGE-NAME PACKAGE-NAME))))
```

(**MENU-PACKAGE-SELECTEDFN**
```
  (IL:LAMBDA (ITEM WINDOW BUTTONS)                                            ; Edited 17-Jul-87 10:13 by DCB
```

;;; check if the new package name is valid and if so initiate the package change by waking up the comand process to handle the command. otherwise
;;; error and reset the package name in the menu to the name of the current package, which is cached on this item.

```
      (LET* ((PACKAGE-NAME-ITEM (IL:LISTGET (IL:FM.ITEMPROP ITEM 'IL:LINKS)
                                        'IL:EDIT))
             (PACKAGE-NAME (IL:FM.ITEMPROP PACKAGE-NAME-ITEM 'IL:LABEL))
             PACKAGE)
            (COND
              ((OR (IL:EQUAL PACKAGE-NAME "")
                   (EQ (CAR BUTTONS)
                       'IL:RIGHT))
               (IL:FM.EDITITEM PACKAGE-NAME-ITEM WINDOW T))
              ((IL:SETQ PACKAGE (FIND-PACKAGE PACKAGE-NAME))
               (IL:FM.ITEMPROP ITEM 'PACKAGE-NAME PACKAGE-NAME)
                (MENU-SELECTEDFN ITEM WINDOW BUTTONS 'SET-PACKAGE (LIST PACKAGE PACKAGE-NAME)))
              (T (IL:|printout| (IL:GETPROMPTWINDOW (IL:MAINWINDOW WINDOW))
                        T "No such package: " PACKAGE-NAME)
                 (IL:FM.CHANGELABEL PACKAGE-NAME-ITEM (IL:FM.ITEMPROP ITEM 'PACKAGE-NAME)
                        WINDOW)))))))
```

(MENU-PRINTBASE-SELECTEDFN
```
  (IL:LAMBDA (ITEM WINDOW BUTTONS)                          ; Edited 17-Jul-87 10:13 by DCB
```

;;;; make sure there is a valid printbase value, and if so, change sedits printbase to it.

```
      (LET* ((PRINTBASE-VALUE-ITEM (IL:LISTGET (IL:FM.ITEMPROP ITEM 'IL:LINKS)
                                                'IL:EDIT))
             (PRINT-BASE (IL:FM.ITEMPROP PRINTBASE-VALUE-ITEM 'IL:STATE)))
            (COND
              ((OR (NULL PRINT-BASE)
                   (EQ (CAR BUTTONS)
                       'IL:RIGHT))
               (IL:FM.EDITITEM PRINTBASE-VALUE-ITEM WINDOW T))
              ((AND (IL:IGREATERP PRINT-BASE 1)
                    (IL:ILEQ PRINT-BASE 36))
               (IL:FM.ITEMPROP ITEM 'PRINTBASE PRINT-BASE)
               (MENU-SELECTEDFN ITEM WINDOW BUTTONS 'SET-PRINT-BASE (LIST PRINT-BASE)))
              (T (IL:|printout| (IL:GETPROMPTWINDOW WINDOW))
                       T "Illegal print-base: " PRINT-BASE)
                 (IL:FM.CHANGESTATE PRINTBASE-VALUE-ITEM (IL:FM.ITEMPROP ITEM 'PRINTBASE)
                       WINDOW))))))
```

(MENU-SELECTEDFN
```
  (IL:LAMBDA (ITEM WINDOW BUTTONS COMMAND EXTRA-ARGS)        ; Edited 17-Jul-87 10:13 by DCB
      (LET ((CONTEXT (IL:WINDOWPROP (IL:MAINWINDOW WINDOW)
                             'EDIT-CONTEXT)))
           (AWAKE-COMMAND-PROCESS CONTEXT (IL:APPEND (LOOKUP-COMMAND (OR COMMAND (IL:FM.ITEMPROP ITEM 'IL:ID)
                                                          (IL:FM.ITEMPROP ITEM 'IL:LABEL))
                                          (IL:FETCH COMMAND-TABLE IL:OF (IL:FETCH ENVIRONMENT
                                                                                  IL:OF CONTEXT)))
                                 EXTRA-ARGS)))))
```

(MENU-SUBSTITUTE-SELECTEDFN
```
  (IL:LAMBDA (ITEM WINDOW BUTTONS)                           ; Edited 17-Jul-87 09:57 by DCB
      (LET ((FIND-ITEM (IL:LISTGET (IL:FM.ITEMPROP ITEM 'IL:LINKS)
                             'FINDITEM))
            (SUBITEM (IL:LISTGET (IL:FM.ITEMPROP ITEM 'IL:LINKS)
                           'IL:EDIT))
            (CONTEXT (IL:WINDOWPROP (IL:MAINWINDOW WINDOW)
                            'EDIT-CONTEXT)))
           (COND
             ((IL:EQUAL (IL:FM.ITEMPROP FIND-ITEM 'IL:LABEL)
                      "")                                    ; need new stuff to find
              (IL:FM.EDITITEM FIND-ITEM WINDOW T))
             ((OR (IL:EQUAL (IL:FM.ITEMPROP SUBITEM 'IL:LABEL)
                         "")
                  (EQ (CAR BUTTONS)
                      'IL:RIGHT))                            ; need new stuff to substitute
              (IL:FM.EDITITEM SUBITEM WINDOW T))
             (T                                              ; call substitute with all the stuff to substitute
                (MENU-SELECTEDFN ITEM WINDOW BUTTONS 'SUBSTITUTE (LIST (IL:FM.ITEMPROP FIND-ITEM 'IL:LABEL)
                                                                      (IL:FM.ITEMPROP SUBITEM 'IL:LABEL)))
                (IL:TTY.PROCESS (IL:WINDOWPROP (IL:MAINWINDOW WINDOW)
                                       'IL:PROCESS)))))))
```

(MUTATE
```
  (IL:LAMBDA (CONTEXT)                                       ; Edited 11-Apr-88 15:58 by woz
      (LET* ((PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
             (SELECTION (IL:|fetch| SELECTION IL:|of| CONTEXT))
             (POINT (IL:|fetch| CARET-POINT IL:|of| CONTEXT))
             (NODE (IL:|fetch| SELECT-NODE IL:|of| SELECTION))
             MUTATOR-STRING MUTATOR RESULT)
            (COND
              ((AND NODE (EQ (IL:|fetch| SELECT-TYPE IL:|of| SELECTION)
                           'STRUCTURE)
                    (NULL (IL:|fetch| SELECT-START IL:|of| SELECTION)))
               (IL:TERPRI PROMPTWINDOW)
               (IL:SETQ MUTATOR-STRING (IL:TTYINPROMPTFORWORD "Mutate by function: " MUTATE-CANDIDATE NIL
                                                PROMPTWINDOW NIL NIL (IL:CHARCODE (IL:CR ^X))))
               (COND
                 ((IL:STRINGP MUTATOR-STRING)
                  (IL:SETQ MUTATOR (IL:NLSETQ (IL:READ (IL:OPENSTRINGSTREAM MUTATOR-STRING 'IL:INPUT))))
                  (IF MUTATOR
                      (IF (DO-MUTATION CONTEXT NODE (CAR MUTATOR))
                          (IL:SETQ MUTATE-CANDIDATE MUTATOR-STRING)
                          (IL:|printout| PROMPTWINDOW T "Error during mutation.  No changes made."))
                      (IL:|printout| PROMPTWINDOW T "Invalid function name: " MUTATOR-STRING)))
                 (T (IL:|printout| PROMPTWINDOW "...aborted"))))
              (T (IL:|printout| PROMPTWINDOW T "Select whole structure to mutate.")))
            T)))
```

(QUOTE-CURRENT-SELECTION
```
  (IL:LAMBDA (CONTEXT CHARCODE QUOTE-TYPE)                   ; Edited 13-Jan-88 13:26 by DCB
```

```
        (CLOSE-OPEN-NODE CONTEXT)
        (LET* ((SELECTION (IL:FETCH SELECTION IL:OF CONTEXT))
               (POINT (IL:FETCH CARET-POINT IL:OF CONTEXT))
               (NODE (IL:FETCH SELECT-NODE IL:OF SELECTION))
               (QUOTE-NODE))
             (WHEN (AND NODE (EQ (IL:FETCH SELECT-TYPE IL:OF SELECTION)
                                 'STRUCTURE))
                 (IL:SETQ QUOTE-NODE (CREATE-QUOTED-GAP BASIC-GAP CONTEXT QUOTE-TYPE))
                 (START-UNDO-BLOCK)
                 (REPLACE-NODE CONTEXT NODE QUOTE-NODE)
                 (REPLACE-NODE CONTEXT (SUBNODE 1 QUOTE-NODE)
                        NODE)
                 (NOTE-CHANGE QUOTE-NODE CONTEXT)
                 (SELECT-NODE CONTEXT QUOTE-NODE)
                 (SET-POINT POINT CONTEXT QUOTE-NODE NIL T NIL 'STRUCTURE)
                 (END-UNDO-BLOCK)))                                        ; must return non-NIL if command executed
        T))
```

(REDISPLAY
```
  (IL:LAMBDA (CONTEXT)                                                     ; Edited  5-Dec-90 14:16 by woz
```

;;; woz: i don't think this function ever gets called!!!

```
    (VERIFY-STRUCTURE CONTEXT NIL NIL T)))
```

(REDO
```
  (IL:LAMBDA (CONTEXT)                                                     ; Edited  7-Jul-87 09:39 by DCB
    (LET ((UNDO-UNDO-LIST (IL:FETCH UNDO-UNDO-LIST IL:OF CONTEXT))
          (PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT)))
         (COND
            (UNDO-UNDO-LIST (SET-SELECTION-NOWHERE (IL:FETCH SELECTION IL:OF CONTEXT))
                   (SET-POINT-NOWHERE (IL:FETCH CARET-POINT IL:OF CONTEXT))
                   (UNDO-EVENT (CAR UNDO-UNDO-LIST)
                          CONTEXT)
                   (IL:REPLACE UNDO-UNDO-LIST IL:OF CONTEXT IL:WITH (CDR UNDO-UNDO-LIST)))
            (T (IL:|printout| PROMPTWINDOW T "No Undo to Undo"))))
    T))
```

(SELECTED-FN-NAME
```
  (IL:LAMBDA (CONTEXT)                                                     ; Edited  7-Jul-87 09:39 by DCB
    (CLOSE-OPEN-NODE CONTEXT)
    (OR (GET-SELECTED-STRUCTURE CONTEXT)
        (LET* ((POINT (IL:FETCH CARET-POINT IL:OF CONTEXT))
               (NODE (IL:FETCH POINT-NODE IL:OF POINT))
               STRUCTURE)
             (WHEN (IL:TYPE? EDIT-NODE NODE)
                 (IL:SETQ STRUCTURE (IL:FETCH STRUCTURE IL:OF NODE))
                 (WHEN (IL:LISTP STRUCTURE)
                     (IL:SETQ STRUCTURE (CAR STRUCTURE)))
                 (WHEN (IL:ATOM STRUCTURE)
                     STRUCTURE))))))
```

(SKIP-TO-GAP
```
  (IL:LAMBDA (CONTEXT)                                                     ; Edited 23-Nov-87 18:19 by DCB
    (LET ((SELECTION (IL:FETCH SELECTION IL:OF CONTEXT))
          (POINT (IL:FETCH CARET-POINT IL:OF CONTEXT))
          (PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT))
          NODE)
         (COND
            ((IL:SETQ NODE (IL:FETCH SELECT-NODE IL:OF SELECTION))
             (UNLESS (SELECT-NEXT-GAP CONTEXT NODE (IL:FETCH SELECT-START IL:OF SELECTION))
                 (IL:|printout| PROMPTWINDOW T "No more blanks to fill in.")))
            ((IL:SETQ NODE (IL:FETCH POINT-NODE IL:OF POINT))
             (UNLESS (SELECT-NEXT-GAP CONTEXT NODE (IF (EQ (IL:FETCH POINT-TYPE IL:OF POINT)
                                                           'STRUCTURE)
                                                       (IL:FETCH POINT-INDEX IL:OF POINT)
                                                       0))
                 (IL:|printout| PROMPTWINDOW T "No more blanks to fill in.")))
            (T (IL:|printout| PROMPTWINDOW T "Select point from which to start search for blanks."))))
    T))
```

(UNDO
```
  (IL:LAMBDA (CONTEXT)                                                     ; Edited  7-Jul-87 09:39 by DCB
    (CLOSE-OPEN-NODE CONTEXT)
    (LET ((UNDO-LIST (IL:FETCH UNDO-LIST IL:OF CONTEXT))
          (PROMPTWINDOW (GET-PROMPT-WINDOW CONTEXT)))
         (COND
            (UNDO-LIST (IL:REPLACE UNDO-LIST IL:OF CONTEXT IL:WITH (IL:FETCH UNDO-UNDO-LIST IL:OF CONTEXT))
                   (SET-SELECTION-NOWHERE (IL:FETCH SELECTION IL:OF CONTEXT))
                   (SET-POINT-NOWHERE (IL:FETCH CARET-POINT IL:OF CONTEXT))
                   (UNDO-EVENT (CAR UNDO-LIST)
                          CONTEXT)
```

```
                        (IL:REPLACE UNDO-UNDO-LIST IL:OF CONTEXT IL:WITH (IL:FETCH UNDO-LIST IL:OF CONTEXT))
                        (WHEN (NULL (IL:REPLACE UNDO-LIST IL:OF CONTEXT IL:WITH (CDR UNDO-LIST)))
                              (IL:REPLACE CHANGED-STRUCTURE? IL:OF CONTEXT IL:WITH NIL)))
                  (T (IL:|printout| PROMPTWINDOW T (IF (IL:FETCH UNDO-UNDO-LIST IL:OF CONTEXT)
                                                       "Nothing else to Undo"
                                                       "Nothing to Undo")))))
       T))
```

(**UNDO-EXTRACT**
   (IL:LAMBDA (CONTEXT NODE SUBNODES)   ; Edited 7-Jul-87 09:39 by DCB

;;; sticks subnodes back into node and revives them.

```
      (RPLACD (IL:FETCH SUB-NODES IL:OF NODE)
              SUBNODES)
      (IL:FOR SUBNODE IL:IN SUBNODES IL:AS INDEX IL:FROM 1 IL:DO (IL:REPLACE SUPER-NODE IL:OF SUBNODE IL:WITH NODE)
                                                                 (IL:REPLACE SUB-NODE-INDEX IL:OF SUBNODE IL:WITH
                                                                                                          INDEX)
                                                                 (DETACH-NODE SUBNODE)
                                                                 (REVIVE-NODE SUBNODE (IL:FETCH DEPTH IL:OF NODE)))
```

   ;; used to reparse here. now if we simply note the change, the format types, format values, and linear forms will be recomputed.

```
      (NOTE-CHANGE NODE CONTEXT)))
```

)

(IL:PUTPROPS **IL:SEDIT-COMMANDS IL:COPYRIGHT** ("Venue & Xerox Corporation" 1986 1987 1988 1990 1991 1992))

FUNCTION INDEX

VARIABLE INDEX

CONSTANT INDEX

PROPERTY INDEX