File created: 13-Jul-2023 14:28:53 **{WMEDLEY}<sources>SEDIT-WINDOW.;6**

edit by: rmk

changes to: (IL:FNS BUTTONEVENTFN)

previous date: 13-Jul-2023 14:06:39 {WMEDLEY}<sources>SEDIT-WINDOW.;5

Read Table: XCL

Package: SEDIT

Format: XCCS


```
(IL:RPAQQ IL:SEDIT-WINDOWCOMS
          ((IL:PROP IL:FILETYPE IL:SEDIT-WINDOW)
           (IL:PROP IL:MAKEFILE-ENVIRONMENT IL:SEDIT-WINDOW)
           (IL:LOCALVARS . T)
           (IL:DECLARE\: IL:DONTCOPY IL:DOEVAL@COMPILE (IL:FILES IL:SEDIT-DECLS))
           (IL:BITMAPS ICON ICON-MASK)
           (IL:VARS ICON-TITLE-REGION (TITLED-ICON (IL:CREATE IL:TITLEDICON IL:ICON IL:_ ICON IL:MASK IL:_
                                                              ICON-MASK IL:TITLEREG IL:_ ICON-TITLE-REGION))
                  (KEEP-WINDOW-REGION T))
           (IL:DECLARE\: IL:DONTCOPY (IL:MACROS IN-TITLE-BAR TRACK-BAR-IN-TRACK-SELECT))
           (IL:FUNCTIONS SELECT-NODE-SEGMENT)
           (IL:FNS BUILD-WINDOW BUTTONEVENTFN CHECK-SELECTION CHECK-SELECTION-SHIFT CLOSEFN
                  CONFLICTING-SELECTION? DISPLAY-SELECTION DRAW-HIGHLIGHT DRAW-OUTLINE DRAW-UNDERLINE EXPANDFN
                  EXPANDREGIONFN EXTEND-SELECTION FINALIZE-MOUSE-SELECTION FIND-LINE-START FIND-NODE
                  GET-DESTINATION-CONTEXT GRAY GROW-CLICK? GROW-SELECTION GROW-SELECTION-DEFAULT
                  HIGHLIGHT-SELECTION ICON-COPYFN LESS-PROMPT-WINDOW NORMALIZE-SELECTION OUTLINE-SELECTION
                  PENDING-DELETE PLACE-CARET-AND-SELECTION PUNT-SET-POINT PUNT-SET-SELECTION REPAINTFN RESHAPEFN
                  SCAN-FOR-BOUNDS SELECT-NODE SELECT-SEGMENT SELECT-SEGMENT-DEFAULT SELECTION-DOWN SELECTION-UP
                  SET-POINT SET-POINT-NOWHERE SET-POINT-UNKNOWN SET-SELECTION SET-SELECTION-ME
                  SET-SELECTION-NOWHERE SHIFT-DOWN SHOW-CARET SHRINKFN STRING-OFFSET TRACK-EXTEND TRACK-SELECT
                  UNDERLINE-SELECTION UPDATE-TITLE)))

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

(IL:PUTPROPS IL:SEDIT-WINDOW 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)
)

(IL:RPAQQ ICON
```


```
)

(IL:RPAQQ ICON-MASK
```

```
)

(IL:RPAQQ ICON-TITLE-REGION (5 16 130 24))

(IL:RPAQ TITLED-ICON (IL:CREATE IL:TITLEDICON IL:ICON IL:_ ICON IL:MASK IL:_ ICON-MASK IL:TITLEREG IL:_
                            ICON-TITLE-REGION))

(IL:RPAQQ KEEP-WINDOW-REGION T)

(IL:DECLARE\: IL:DONTCOPY

(IL:DECLARE\: IL:EVAL@COMPILE

(IL:PUTPROPS IN-TITLE-BAR IL:MACRO ((WINDOW)
                                    (NOT (IL:INSIDEP (IL:DSPCLIPPINGREGION NIL WINDOW)
                                                     (IL:LASTMOUSEX WINDOW)
                                                     (IL:LASTMOUSEY WINDOW)))))

(IL:PUTPROPS TRACK-BAR-IN-TRACK-SELECT IL:MACRO (NIL (WHEN (OR (IL:NEQ POINT? (IL:|fetch| POINT-TYPE IL:|of|
```

```
                                                                    PENDING-CARET
                                                                ))
                                        (IL:NEQ BAR-X (IL:|fetch| POINT-X IL:|of|
                                                                    PENDING-CARET
                                                                ))
                                        (IL:NEQ BAR-LINE (IL:|fetch| POINT-LINE
                                                           IL:|of| PENDING-CARET)))
                                    (WHEN POINT?
                                        (IL:BLTSHADE IL:BLACKSHADE WINDOW BAR-X BAR-Y 1
                                              BAR-HEIGHT 'IL:INVERT))
                                    (WHEN (IL:SETQ POINT? (IL:|fetch| POINT-TYPE IL:|of|
                                                                    PENDING-CARET
                                                                ))
                                        (IL:SETQ BAR-X (IL:|fetch| POINT-X IL:|of|
                                                                    PENDING-CARET
                                                                ))
                                        (IL:SETQ BAR-LINE (IL:|fetch| POINT-LINE IL:|of|
                                                                    PENDING-CARET
                                                                ))
                                        (IL:SETQ BAR-HEIGHT (IL:IPLUS (IL:|fetch| LINE-ASCENT
                                                                        IL:|of| BAR-LINE)
                                                                    (IL:|fetch| LINE-DESCENT
                                                                        IL:|of| BAR-LINE)))
                                        (IL:SETQ BAR-Y (IL:IDIFFERENCE (IL:|fetch| YCOORD
                                                                         IL:|of| BAR-LINE)
                                                                    (IL:IPLUS (IL:|fetch| LINE-SKIP
                                                                                IL:|of| BAR-LINE)
                                                                        BAR-HEIGHT)))
                                        (IL:BLTSHADE IL:BLACKSHADE WINDOW BAR-X BAR-Y 1
                                              BAR-HEIGHT 'IL:INVERT)))))
)
)
```

```
(DEFUN SELECT-NODE-SEGMENT (CONTEXT NODE &OPTIONAL (START 1)
                                    END)
```

;;; set the current selection to be a segment under this node

```
   (LET ((SELECTION (IL:FETCH SELECTION IL:OF CONTEXT))
         (POINT (IL:FETCH CARET-POINT IL:OF CONTEXT)))
        (IL:|replace| SELECT-NODE IL:|of| SELECTION IL:|with| NODE)
        (IL:|replace| SELECT-START IL:|of| SELECTION IL:|with| START)
        (IL:|replace| SELECT-END IL:|of| SELECTION IL:|with| END)
        (SELECT-SEGMENT SELECTION CONTEXT NODE)
```

;; set point to be the selection. This should really be done by select-segment but it doesn't because it expects finalization code to be run after
;; it cause it's generally called from the mouse tracking code which finalizes.

```
        (PENDING-DELETE POINT SELECTION)))
```

```
(IL:DEFINEQ
```

(BUILD-WINDOW

```
  (IL:LAMBDA (CONTEXT)                                    ; Edited  2-Apr-92 10:59 by jds
```

;;; create a new window to edit in. called from setup.new.context when an sedit is started.

```
    (LET ((ENVIRONMENT (IL:|fetch| ENVIRONMENT IL:|of| CONTEXT))
          (DISPLAY-WINDOW (IL:CREATEW (LESS-PROMPT-WINDOW (GET-WINDOW-REGION CONTEXT :CREATE
                                                                (IL:|fetch| ICON-TITLE IL:|of| CONTEXT)
                                                                (IL:|fetch| EDIT-TYPE IL:|of| CONTEXT))
                                                          IL:DEFAULTFONT)
                                      (IL:CONCAT EDITOR-NAME " parsing " (OR (IL:|fetch| ICON-TITLE IL:|of| CONTEXT)
                                                                             ""))))))
         (IL:WINDOWPROP DISPLAY-WINDOW 'EDIT-CONTEXT CONTEXT)
         (IL:WINDOWPROP DISPLAY-WINDOW 'IL:SCROLLEXTENTUSE '(- . +))
         (IL:WINDOWPROP DISPLAY-WINDOW 'IL:WINDOWENTRYFN (IL:FUNCTION BUTTONEVENTFN))
         (IL:WINDOWPROP DISPLAY-WINDOW 'IL:BUTTONEVENTFN (IL:FUNCTION BUTTONEVENTFN))
         (IL:WINDOWPROP DISPLAY-WINDOW 'IL:RIGHTBUTTONFN (IL:FUNCTION BUTTONEVENTFN))
         (IL:WINDOWPROP DISPLAY-WINDOW 'IL:EXPANDREGIONFN (IL:FUNCTION EXPANDREGIONFN))
         (IL:WINDOWPROP DISPLAY-WINDOW 'IL:CLOSEFN (IL:FUNCTION CLOSEFN))
         (IL:WINDOWPROP DISPLAY-WINDOW 'IL:SHRINKFN (IL:FUNCTION SHRINKFN))
         (IL:WINDOWPROP DISPLAY-WINDOW 'IL:EXPANDFN (IL:FUNCTION EXPANDFN))
         (IL:WINDOWPROP DISPLAY-WINDOW 'IL:RESHAPEFN (IL:FUNCTION RESHAPEFN))
```

;; get the prompt window after setting up all the window fn, so the'll be in the proper order

```
         (IL:GETPROMPTWINDOW DISPLAY-WINDOW 1 IL:DEFAULTFONT)
         (IL:|replace| DISPLAY-WINDOW IL:|of| CONTEXT IL:|with| DISPLAY-WINDOW)
         (IL:WYOFFSET (IL:SUB1 (IL:WINDOWPROP DISPLAY-WINDOW 'IL:HEIGHT))
               DISPLAY-WINDOW)
```

;; These window fns go AFTER the promptwindow setup, so we don't try to repaint the window in the course of adding the prompt window.
;; This fixes AR 11376

```
         (IL:WINDOWPROP DISPLAY-WINDOW 'IL:REPAINTFN (IL:FUNCTION REPAINTFN))
         (IL:WINDOWPROP DISPLAY-WINDOW 'IL:SCROLLFN (IL:FUNCTION IL:SCROLLBYREPAINTFN))
         (IL:|replace| WINDOW-LEFT IL:|of| CONTEXT IL:|with| (IL:|fetch| (IL:REGION IL:LEFT) IL:|of| (IL:DSPCLIPPINGREGION
                                                                                                      NIL DISPLAY-WINDOW)))
```

```
        (IL:|replace| WINDOW-BOTTOM IL:|of| CONTEXT IL:|with| (IL:|fetch| (IL:REGION IL:BOTTOM) IL:|of| (IL:DSPCLIPPINGREGION
                                                                                                        NIL DISPLAY-WINDOW))
              )
        (IL:|replace| WINDOW-RIGHT IL:|of| CONTEXT IL:|with| (IL:|fetch| (IL:REGION IL:RIGHT) IL:|of| (IL:DSPCLIPPINGREGION
                                                                                                    NIL DISPLAY-WINDOW)))
        (IL:|replace| WINDOW-TOP IL:|of| CONTEXT IL:|with| (IL:|fetch| (IL:REGION IL:TOP) IL:|of| (IL:DSPCLIPPINGREGION NIL
                                                                                                     DISPLAY-WINDOW)))
        (IL:DSPLINEFEED (IL:IMINUS (IL:IPLUS (IL:FONTPROP (IL:|fetch| DEFAULT-FONT IL:|of| ENVIRONMENT)
                                                          'IL:HEIGHT)
                                              (IL:|fetch| DEFAULT-LINE-SKIP IL:|of| ENVIRONMENT)))
              DISPLAY-WINDOW)
        ;; set the window's right margin big enough that things won't be wrapped on us.  this is sort of gross -- there should be a way to completely
        ;; disable wrap
        (IL:DSPRIGHTMARGIN 64000 DISPLAY-WINDOW))))
```

(**BUTTONEVENTFN**

```
(IL:LAMBDA (WINDOW)                                                    ; Edited 13-Jul-2023 14:27 by rmk
                                                                       ; Edited 20-Jun-2023 21:10 by rmk
                                                                       ; Edited 17-Jun-2023 19:59 by rmk
                                                                       ; Edited 23-Apr-2018 09:37 by rmk:
    ;; called by the window system whenever the user hits a mouse button in an SEdit window.  allows selection and setting the caret point
    (LET* ((CONTEXT (IL:WINDOWPROP WINDOW 'EDIT-CONTEXT))
           (LOCK (AND CONTEXT (IL:|fetch| CONTEXT-LOCK IL:|of| CONTEXT)))
           (SHIFT-DOWN (SHIFT-DOWN)))
          (COND
             ((IL:LASTMOUSESTATE IL:UP)
              ;; oops, no mouse buttons down.  what are we doing here?
              NIL)
             ((NOT (AND CONTEXT (IL:WINDOWPROP WINDOW 'IL:PROCESS)))
              ;; this context or process is dead.  make it a dead SEdit.
              (IL:|printout| (IL:GETPROMPTWINDOW WINDOW)
                     T "This SEdit is dead.")
              (IL:WINDOWPROP WINDOW 'IL:REPAINTFN NIL)
              (IL:WINDOWPROP WINDOW 'IL:RESHAPEFN 'IL:DON\'T)
              (IL:WINDOWPROP WINDOW 'IL:SHRINKFN 'IL:DON\'T)
              (AND (IL:LASTMOUSESTATE IL:RIGHT)
                   (IL:DOWINDOWCOM WINDOW)))
             ((AND (IL:LASTMOUSESTATE IL:RIGHT)
                   (IN-TITLE-BAR WINDOW))
              ;; right buttoning the title bar or window border gives the default menu of window commands.  Not interlocked because want to be
              ;; able to move window under a break that has the lock.
              (IL:\\CARET.DOWN)
              (IL:DOWINDOWCOM WINDOW))
             ((AND (NOT (IL:TTY.PROCESSP (IL:WINDOWPROP WINDOW 'IL:PROCESS)))
                   (NOT SHIFT-DOWN))
              ;; just grab the tty and don't change state
              (IL:TOTOPW WINDOW)
              (IL:TTY.PROCESS (IL:WINDOWPROP WINDOW 'IL:PROCESS)))
             ((AND (EQ SHIFT-DOWN 'COPY)
                   (IL:MOUSESTATE IL:LEFT)
                   (IN-TITLE-BAR WINDOW))
              ;; RMK: copy-select in the title bar: return the thing being edited. Previous attempt was too immediate, did not conform to usual
              ;; mouse-up conventions.
              (IL:WHILE (EQ 'COPY (SHIFT-DOWN)))
              (IL:GETMOUSESTATE)
              (WHEN (IN-TITLE-BAR WINDOW)
                  (LET ((NAME (IL:LISTGET (IL:WINDOWPROP WINDOW 'TITLE-INFO)
                                          :|name|)))
                       (WHEN NAME                                      ; Not sure about FLG and RDTBL
                           (IL:COPYINSERT NAME)))))
             ((OR (EQ SHIFT-DOWN 'COPY)
                  (IL:OBTAIN.MONITORLOCK LOCK T))
              ;; at this point we must have the lock, unless we're shift selecting (Copy only: Move and Delete are non-passive operation and must
              ;; lock)
              (IL:\\CARET.DOWN)
              (IL:TOTOPW WINDOW)
              (COND
                 ((AND (IN-TITLE-BAR WINDOW)
                       (OR (IL:LASTMOUSESTATE IL:MIDDLE)
                           (AND (IL:LASTMOUSESTATE IL:LEFT)
                                (IL:KEYDOWNP 'IL:CTRL))))
                  ;; popup help command menu here.
                  ;; RMK:  CTRL-LEFT = MIDDLE
                  (HELPMENU CONTEXT))
                 (T (WITH-PROFILE (IL:|fetch| PROFILE IL:|of| CONTEXT)
                           (PROG NIL
                                 (CLOSE-OPEN-NODE CONTEXT)
```

```
                    ;; record that we're busy making a selection in this window, and make sure that variables we use for recording our
                    ;; temporary state are all ready for action.  note that these are global vars, and hence all this code is nonrentrant.
                    ;; shouldn't be a problem, since there's only one mouse
                         (IL:SETQ SELECTION-PENDING? CONTEXT)
                         (IL:SETQ PENDING-LAST-X (IL:|fetch| LAST-MOUSE-X IL:|of| CONTEXT))
                         (IL:SETQ PENDING-LAST-Y (IL:|fetch| LAST-MOUSE-Y IL:|of| CONTEXT))
                         (IL:SETQ PENDING-TYPE (IL:|fetch| LAST-MOUSE-TYPE IL:|of| CONTEXT))
                         (IL:SETQ PENDING-SHIFT SHIFT-DOWN)
                         (IL:|replace| SELECT-NODE IL:|of| PENDING-SELECTION IL:|with| NIL)
                         (WHEN (NOT PENDING-SHIFT)
                              ;; if they're setting a new selection take down the main selection
                              (SELECTION-DOWN CONTEXT))
                         (IL:SETQ LAST-MOVE-CLOCK NIL)
                         (IL:SETQ BUTTON-STRING-NODE NIL)
                     MOUSE-BUTTON-DOWN
                         (IF (IL:LASTMOUSESTATE IL:RIGHT)
                             (TRACK-EXTEND CONTEXT WINDOW)
                             (TRACK-SELECT CONTEXT WINDOW))
                         (IL:|until| (CHECK-SELECTION-SHIFT CONTEXT T)
                            IL:|do| (WHEN (NOT (IL:MOUSESTATE IL:UP))
                                          (GO MOUSE-BUTTON-DOWN))
                                    (WHEN (IL:IN/SCROLL/BAR? WINDOW IL:LASTMOUSEX IL:LASTMOUSEY)
                                                                  ; let them scroll while making a selection
                                        (IL:SCROLL.HANDLER WINDOW))
                                    (IL:BLOCK))
                         (IL:SETQ SELECTION-PENDING? NIL)
                                                                  ; figure out what we should do
                         (FINALIZE-MOUSE-SELECTION CONTEXT WINDOW)))))
                (OR (EQ SHIFT-DOWN 'COPY)
                    (IL:RELEASE.MONITORLOCK LOCK)))))))
```

(CHECK-SELECTION

```
  (IL:LAMBDA (SELECTION POINT)                                    ; Edited 27-Jun-88 15:47 by woz
```

;;; called from update each time through. check the selection for dead node, and for pending delete inconsistency.

```
    (LET ((NODE (IL:|fetch| SELECT-NODE IL:|of| SELECTION))
          (START (IL:|fetch| SELECT-START IL:|of| SELECTION))
          (END (IL:|fetch| SELECT-END IL:|of| SELECTION))
          SUBNODE)
         (WHEN (AND NODE (DEAD-NODE? NODE))
               (IL:REPLACE SELECT-NODE IL:OF SELECTION IL:WITH NIL))
         (COND
            ((EQ (IL:|fetch| POINT-NODE IL:|of| POINT)
                 SELECTION)
             (COND
                ((NULL NODE)
                 (IL:REPLACE POINT-NODE IL:OF POINT IL:WITH NIL))
                ((NOT (IL:FETCH PENDING-DELETE? IL:OF SELECTION))
                 (IL:SHOULDNT "pending delete inconsistency"))))
            ((AND NODE (IL:|fetch| PENDING-DELETE? IL:|of| SELECTION))
             (IL:SHOULDNT "pending delete inconsistency")))
```

;; try to simplify the selection. if it's a single node structure segment (single subnode selected), select the subnode directly instead.

```
         (WHEN (AND NODE (EQ (IL:|fetch| SELECT-TYPE IL:|of| SELECTION)
                             'STRUCTURE)
                     (NOT (IL:|fetch| PENDING-DELETE? IL:|of| SELECTION))
                     START
                     (OR (NULL END)
                         (EQL START END))
                     (IL:|type?| EDIT-NODE (SETQ SUBNODE (NTH START (IL:FETCH SUB-NODES IL:OF NODE)))))
               (IL:|replace| SELECT-NODE IL:|of| SELECTION IL:|with| SUBNODE)
               (IL:|replace| SELECT-START IL:|of| SELECTION IL:|with| NIL)
               (IL:|replace| SELECT-END IL:|of| SELECTION IL:|with| NIL)))))
```

(CHECK-SELECTION-SHIFT

```
  (IL:LAMBDA (CONTEXT LET-GO)                                     ; Edited  7-Jul-87 13:00 by DCB
```

;;; check for modifier keys being held down during this selection and update the display if they have changed. if let.go is true, and there are no modifier
;;; keys down, the selection is completed and return T to wake up the buttoneventfn

```
    (LET ((NEW-SHIFT (SHIFT-DOWN)))
         (COND
            ((AND LET-GO (NULL NEW-SHIFT))                        ; no mouse buttons, and no modifier keys -- we're done
             T)
            (T (WHEN (IL:NEQ NEW-SHIFT PENDING-SHIFT)
                     (IL:\\CARET.DOWN)
                     (WHEN (EQ PENDING-SHIFT 'MOVE)
                          ;; since move selection requires two keys (at least on my keyboard) we give it a little hysteresis so you don't have to
                          ;; release both keys at *exactly* the same time
                          (IL:SETQ LAST-MOVE-CLOCK (IL:CLOCK 0)))  ; change the selection display
                     (DISPLAY-SELECTION PENDING-SELECTION (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
```

```
                    PENDING-SHIFT)
                (DISPLAY-SELECTION PENDING-SELECTION (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                    NEW-SHIFT)                                  ; make the new shift type current
                (IL:SETQ PENDING-SHIFT NEW-SHIFT))
            NIL)))))
```

(CLOSEFN

```
(IL:LAMBDA (WINDOW)                                             ; Edited 5-Dec-90 18:07 by woz
```

;;; to be called by the window system when SEdit windows are closed. if there's a process, wake it up with a complete command. otherwise just trash ;;; the context. grab the lock here, because it wasn't yet grabbed by the buttoneventfn.

```
    (LET ((CONTEXT (IL:WINDOWPROP WINDOW 'EDIT-CONTEXT)))
         (WHEN CONTEXT
             (COND
                ((IL:OBTAIN.MONITORLOCK (IL:|fetch| CONTEXT-LOCK IL:|of| CONTEXT)
                        T)
                 (IL:RELEASE.MONITORLOCK (IL:|fetch| CONTEXT-LOCK IL:|of| CONTEXT))
                                                                ; release before waking sedit
                 ;; if there's a stupid attached menu, close it first so we'll release the correct region
                 (WHEN (IL:WINDOWPROP WINDOW 'MENU)
                     (IL:CLOSEW (IL:WINDOWPROP WINDOW 'MENU)))
                 (COND
                    ((IL:WINDOWPROP WINDOW 'IL:PROCESS)
                     (COND
                        ((EQ (IL:PROCESSPROP (IL:THIS.PROCESS)
                                    'IL:NAME)
                             'IL:MOUSE)
                         ;; if we're running under the mouse, just wake up the SEdit process and let it close the window.  That way all
                         ;; completion happens under the command process, not under the mouse.
                         (AWAKE-COMMAND-PROCESS CONTEXT '(COMPLETE NIL :CLOSE))
                         'IL:DON\'T)
                        (T (SAVE-WINDOW-REGION CONTEXT :CLOSE (IL:|fetch| ICON-TITLE IL:|of| CONTEXT)
                                  (IL:|fetch| EDIT-TYPE IL:|of| CONTEXT)
                                  (IL:WINDOWREGION WINDOW)))))
                    (T ;; We take this branch when an sedit icon is closed.  The process is already dead, but we still have the context to junk.
                       ;; Also, This case CAN HAPPEN IF SOMEBODY RETFROMs sedit or some process involved in cleanup gets an error so
                       ;; the sedit dies.
                       (SAVE-WINDOW-REGION CONTEXT :CLOSE-ICON (AND CONTEXT (IL:|fetch| ICON-TITLE IL:|of| CONTEXT))
                              (AND CONTEXT (IL:|fetch| EDIT-TYPE IL:|of| CONTEXT))
                              (IL:WINDOWREGION WINDOW))
                       (DISINTEGRATE-CONTEXT CONTEXT))))
                (T (FORMAT (GET-PROMPT-WINDOW CONTEXT)
                           "~%Can't close.  SEdit is busy")
                   'IL:DON\'T))))))
```

(CONFLICTING-SELECTION?

```
(IL:LAMBDA (CONTEXT DESTINATION-CONTEXT)                        ; Edited 7-Jul-87 13:00 by DCB
```

;; determine if the pending selection conflicts with the main selection in context. there is a conflict for pending selections which get deleted (Delete ;; or Move) because the deletion can mess up the main selection. In the case of Move, if the destination is the same SEdit and the main selection ;; is pending delete, then this parks the point for the move, so leave it up; the copy meshod will worry about overlaps.

```
    (LET ((SELECTION (IL:FETCH SELECTION IL:OF CONTEXT)))
         (WHEN (IL:FETCH SELECT-NODE IL:OF SELECTION)
             (OR (EQ PENDING-SHIFT 'DELETE)
                 (AND (EQ PENDING-SHIFT 'MOVE)
                      (IL:NEQ CONTEXT DESTINATION-CONTEXT)))))))
```

(DISPLAY-SELECTION

```
(IL:LAMBDA (SELECTION WINDOW TYPE)                              ; Edited 7-Jul-87 13:01 by DCB
```

;;; display the current selection with the appropriate markings (outline or underline, gray or black)

```
    (WHEN (IL:FETCH SELECT-NODE IL:OF SELECTION)
        (COND
           ((DEAD-NODE? (IL:FETCH SELECT-NODE IL:OF SELECTION))
            (IL:REPLACE SELECT-NODE IL:OF SELECTION IL:WITH NIL))
           (T (IL:SELECTQ TYPE
                    (NIL ;; normal selection -- black underline, or pending delete selection -- black outline
                         (IF (IL:FETCH PENDING-DELETE? IL:OF SELECTION)
                             (OUTLINE-SELECTION SELECTION WINDOW IL:BLACKSHADE)
                             (UNDERLINE-SELECTION SELECTION WINDOW IL:BLACKSHADE)))
                    (COPY                                       ; copy selection -- gray underline
                          (UNDERLINE-SELECTION SELECTION WINDOW (GRAY WINDOW)))
                    (MOVE                                       ; move selection -- gray outline
                          (OUTLINE-SELECTION SELECTION WINDOW (GRAY WINDOW)))
                    (DELETE                                     ; delete selection -- inverted
                            (HIGHLIGHT-SELECTION SELECTION WINDOW IL:BLACKSHADE))
                    (IL:SHOULDNT "unknown selection display type"))
              T)))))
```

(DRAW-HIGHLIGHT

```
  (IL:LAMBDA (X-1 X-2 X-3 W Y-1 H-1 Y-2 H-2 WINDOW SHADE)        ; Edited 17-Nov-87 11:21 by DCB
```

;;; inverts the selection. x1 is the left edge of the region, x2 is the left edge of the first line (which may be indented) x3 is right edge of the last line, w is ;;; the width, y1 is the top, h1 is the height of the first line, y2 is the top of the last line, and h2 is its height. the region will be painted with the specified ;;; shade in invert mode

```
    (IL:SETQ X-3 (IL:ADD1 X-3))
    (IL:SETQ W (IL:ADD1 W))
    (COND
       ((EQ (IL:SETQ Y-1 (IL:ADD1 Y-1))
            (IL:SETQ Y-2 (IL:ADD1 Y-2)))
        (IL:BLTSHADE SHADE WINDOW X-2 (IL:IDIFFERENCE Y-1 H-1)
               (IL:IDIFFERENCE X-3 X-2)
               H-1
               'IL:INVERT))
       (T (WHEN (IL:NEQ X-1 X-2)
              (IL:SETQ Y-1 (IL:IDIFFERENCE Y-1 H-1))
              (IL:BLTSHADE SHADE WINDOW X-2 Y-1 (IL:IDIFFERENCE (IL:IPLUS X-1 W)
                                                        X-2)
                     H-1
                     'IL:INVERT))
          (IF (IL:NEQ X-3 (IL:IPLUS X-1 W))
              (IL:BLTSHADE SHADE WINDOW X-1 (IL:IDIFFERENCE Y-2 H-2)
                     (IL:IDIFFERENCE X-3 X-1)
                     H-2
                     'IL:INVERT)
              (IL:SETQ Y-2 (IL:IDIFFERENCE Y-2 H-2)))
          (IL:BLTSHADE SHADE WINDOW X-1 Y-2 W (IL:IDIFFERENCE Y-1 Y-2)
                 'IL:INVERT)))))
```

(DRAW-OUTLINE

```
  (IL:LAMBDA (X-1 X-2 X-3 W Y-1 H-1 Y-2 H-2 WINDOW SHADE)        ; Edited 17-Nov-87 11:21 by DCB
```

;;; outline the selection. arguments are the same as draw.highlight. the selection will be surrounded by a 1 pixel wide border in the specified shade

```
    (IL:SETQ H-1 (IL:IDIFFERENCE Y-1 H-1))
    (IL:SETQ H-2 (IL:IDIFFERENCE Y-2 H-2))
    (IL:SETQ W (IL:IPLUS X-1 W))
    (WHEN (EQ Y-1 Y-2)
        (IL:SETQ X-1 X-2)
        (IL:SETQ W X-3))
    (COND
       ((EQ X-1 X-2)
        (IL:BLTSHADE SHADE WINDOW (IL:SUB1 X-1)
               H-2 1 (IL:IDIFFERENCE Y-1 H-2)
               'IL:INVERT))
       (T (IL:BLTSHADE SHADE WINDOW (IL:SUB1 X-1)
                 H-2 1 (IL:IDIFFERENCE H-1 H-2)
                 'IL:INVERT)
          (IL:BLTSHADE SHADE WINDOW (IL:SUB1 X-1)
                 H-1
                 (IL:IDIFFERENCE X-2 X-1)
                 1
                 'IL:INVERT)
          (IL:BLTSHADE SHADE WINDOW (IL:SUB1 X-2)
                 H-1 1 (IL:IDIFFERENCE Y-1 H-1)
                 'IL:INVERT)))
    (IL:BLTSHADE SHADE WINDOW (IL:SUB1 X-2)
           Y-1
           (IL:IDIFFERENCE (IL:IPLUS 2 W)
                  X-2)
           1
           'IL:INVERT)
    (IL:BLTSHADE SHADE WINDOW X-1 H-2 (IL:IDIFFERENCE X-3 X-1)
           1
           'IL:INVERT)
    (COND
       ((EQ X-3 W)
        (IL:BLTSHADE SHADE WINDOW X-3 H-2 1 (IL:IDIFFERENCE Y-1 H-2)
               'IL:INVERT))
       (T (IL:BLTSHADE SHADE WINDOW X-3 H-2 1 (IL:IDIFFERENCE Y-2 H-2)
                 'IL:INVERT)
          (IL:BLTSHADE SHADE WINDOW X-3 Y-2 (IL:IDIFFERENCE W X-3)
                 1
                 'IL:INVERT)
          (IL:BLTSHADE SHADE WINDOW W Y-2 1 (IL:IDIFFERENCE Y-1 Y-2)
                 'IL:INVERT)))))
```

(DRAW-UNDERLINE

```
  (IL:LAMBDA (STARTX FIRST ENDX LAST WINDOW SHADE)               ; Edited 17-Jul-87 10:10 by DCB
```

;;; underline the selection. first and last are the first and last lines, and startx and endx are the x coordinates of the ends of the selection on those lines. ;;; the selection will be underlined with a 2 pixel wide line of the specified shade

```
    (IL:UNTIL (EQ FIRST LAST) IL:DO (IL:BLTSHADE SHADE WINDOW STARTX (IL:FETCH NEXT-LINE-Y IL:OF FIRST)
                                           (IL:IDIFFERENCE (IL:FETCH LINE-LENGTH IL:OF FIRST)
                                                  STARTX)
                                           2
                                           'IL:INVERT)
                                    (IL:SETQ FIRST (CAR (IL:FETCH NEXT-LINE IL:OF FIRST)))
                                    (IL:SETQ STARTX (IL:FETCH INDENT IL:OF FIRST)))
    (IL:BLTSHADE SHADE WINDOW STARTX (IL:FETCH NEXT-LINE-Y IL:OF FIRST)
           (IL:IDIFFERENCE ENDX STARTX)
           2
           'IL:INVERT)))
```

(EXPANDFN

```
  (IL:LAMBDA (WINDOW)                                                    ; Edited 19-Aug-87 15:39 by drc:
```

;;; called by the window system when SEdit window icons are expanded. start a new command process for the window

```
    (LET ((CONTEXT (IL:WINDOWPROP WINDOW 'EDIT-CONTEXT)))
         (WHEN (NOT (IL:WINDOWPROP WINDOW 'IL:PROCESS))
             (IL:|replace| EVAL-IN-PROCESS IL:|of| CONTEXT IL:|with| (EVAL-IN-PROCESS))
             (START-PROCESS CONTEXT)))))
```

(EXPANDREGIONFN

```
  (IL:LAMBDA (WINDOW)                                                    ; Edited  8-Jan-88 17:49 by woz
```

;;; calculates a new region for this window as it is expanded. Return NIL if don't want to reshape the window. remember the region manager gives a
;;; region including the prompt window, so subtract it before handing the region to the main window.

```
    (LET* ((CONTEXT (IL:WINDOWPROP WINDOW 'EDIT-CONTEXT))
           (REGION (GET-WINDOW-REGION CONTEXT :EXPAND (IL:|fetch| ICON-TITLE IL:|of| CONTEXT)
                          (IL:|fetch| EDIT-TYPE IL:|of| CONTEXT))))
          (AND REGION (LESS-PROMPT-WINDOW REGION IL:DEFAULTFONT)))))
```

(EXTEND-SELECTION

```
  (IL:LAMBDA (SELECTION CONTEXT X Y)                                     ; Edited 24-Nov-87 09:53 by DCB
```

;;; expand the given selection to include the point (x,y)

```
    (LET (NODE INDEX OFFSET LINE LINEAR)
         (WHEN (AND (IL:INSIDEP (IL:DSPCLIPPINGREGION NIL (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT))
                           X Y)
                    (IL:SETQ LINE (FIND-LINE-START Y CONTEXT))
                    (IL:SETQ LINEAR (FIND-NODE X LINE CONTEXT)))
             ;; we've found the linear item they're pointing at.  figure out what node it belongs to, what its index in the linear form is, and how far into
             ;; the item the position is
             (IL:SETQ NODE (IL:FETCH DESTINATION IL:OF (CDR (LAST LINEAR))))
             (IL:SETQ INDEX (IL:FOR I IL:FROM 1 IL:AS (X IL:_ (IL:FETCH LINEAR-FORM IL:OF NODE))
                               IL:BY (CDR X) IL:THEREIS (EQ X LINEAR)))
             (IL:SETQ OFFSET (IF (IL:TYPE? LINE-START (CAR LINEAR))
                                 (IF (EQ 0 (IL:FETCH \\X IL:OF CONTEXT))
                                     1
                                     -1)
                                 (IL:IDIFFERENCE X (IL:FETCH \\X IL:OF CONTEXT))))
             (COND
                ((AND (IL:FETCH SELECT-START IL:OF SELECTION)
                      (EQ (IL:FETCH SELECT-NODE IL:OF SELECTION)
                          NODE))
                 ;; easy case -- the current selection's node is the one to handle it
                 (SELECT-SEGMENT SELECTION CONTEXT NODE NIL INDEX OFFSET (CAR LINEAR)))
                (T ;; harder.  we've got to figure out the lowest common subnode and get it to do the work.  this could (and should) be simplified
                   ;; and sped up now that we store depth information.  its is currently so ugly that it's not even worth trying to explain
                   (PROG ((A (IL:FETCH SELECT-NODE IL:OF SELECTION))
                          (B NODE)
                          T-0 T-1 T-2)
                     LOOPB
                         (WHEN (NOT (IL:FETCH SUPER-NODE IL:OF A))
                               (GO LOOPA))
                         (IL:SETQ T-2 A)
                         (IL:SETQ A (IL:FETCH SUPER-NODE IL:OF A))
                         (IL:SETQ T-1 NODE)
                         (IL:SETQ T-0 T-1)
                     LOOPB-2
                         (WHEN (EQ T-0 A)
                               (GO DONE))
                         (WHEN (EQ T-0 B)
                               (GO LOOPA))
                         (IL:SETQ T-1 T-0)
                         (IL:SETQ T-0 (IL:FETCH SUPER-NODE IL:OF T-0))
                         (GO LOOPB-2)
                     LOOPA
```

```
                    (WHEN (NOT (IL:FETCH SUPER-NODE IL:OF B))
                          (GO LOOPB))
                    (IL:SETQ T-2 B)
                    (IL:SETQ B (IL:FETCH SUPER-NODE IL:OF B))
                    (IL:SETQ T-1 (IL:FETCH SELECT-NODE IL:OF SELECTION))
                    (IL:SETQ T-0 T-1)
                LOOPA-2
                    (WHEN (EQ T-0 B)
                          (GO DONE))
                    (WHEN (EQ T-0 A)
                          (GO LOOPB))
                    (IL:SETQ T-1 T-0)
                    (IL:SETQ T-0 (IL:FETCH SUPER-NODE IL:OF T-0))
                    (GO LOOPA-2)
                DONE
                    (COND
                       ((EQ (IL:FETCH SELECT-NODE IL:OF SELECTION)
                            T-0)
                        (IF (IL:FETCH SELECT-START IL:OF SELECTION)
                            (SELECT-SEGMENT SELECTION CONTEXT T-0 T-2 NIL OFFSET (CAR LINEAR))
                            (SELECT-SEGMENT SELECTION CONTEXT (IL:FETCH SUPER-NODE IL:OF T-0)
                                   T-0 T-0 NIL OFFSET (CAR LINEAR))))
                       ((EQ NODE T-0)
                        (SELECT-SEGMENT SELECTION CONTEXT NODE T-2 INDEX OFFSET (CAR LINEAR)))
                       (T (SELECT-SEGMENT SELECTION CONTEXT T-0 T-1 T-2 NIL OFFSET (CAR LINEAR)))))))))))
```

(FINALIZE-MOUSE-SELECTION

```
(IL:LAMBDA (CONTEXT WINDOW)                                    ; Edited 7-Jul-87 13:03 by DCB
```

;; all mouse buttons and modifier keys have been released, so the selection's completed. figure out just what it was that was selected, and if it's a ;; copy, move, or delete, do it

```
    (LET ((SELECTION (IL:|fetch| SELECTION IL:|of| CONTEXT)))
         (COND
            (PENDING-SHIFT                                     ; some action required
                   (WHEN (IL:|fetch| SELECT-NODE IL:|of| PENDING-SELECTION)
                       (LET ((DESTINATION-CONTEXT (GET-DESTINATION-CONTEXT))
                             DESTINATION-POINT)
                            (IL:\\CARET.DOWN)                  ; need this here because get.destination.context lets the caret
                                                               ; flash again.
                            (WHEN (IL:NEQ PENDING-SHIFT 'COPY) ; for Move or Delete
                                (SELECTION-DOWN CONTEXT))
```

;; take down the pending (shift) selection

```
                            (DISPLAY-SELECTION PENDING-SELECTION WINDOW PENDING-SHIFT)
                            (WHEN (AND LAST-MOVE-CLOCK (IL:ILESSP (IL:CLOCK 0)
                                                                  (IL:IPLUS LAST-MOVE-CLOCK 250)))
```

;; if they release the two keys within a quarter second, we'll assume it was a move

```
                                (IL:SETQ PENDING-SHIFT 'MOVE))
                            (WHEN (CONFLICTING-SELECTION? CONTEXT DESTINATION-CONTEXT)
```

;; if the selection conflicts then waste it.

```
                                (SET-SELECTION-NOWHERE SELECTION))
                            (COND
                               ((EQ PENDING-SHIFT 'DELETE)
                                (DELETE-NODES (IL:|fetch| SELECT-NODE IL:|of| PENDING-SELECTION)
                                       CONTEXT
                                       (IL:|fetch| SELECT-START IL:|of| PENDING-SELECTION)
                                       (IL:|fetch| SELECT-END IL:|of| PENDING-SELECTION)
                                       (IL:|fetch| CARET-POINT IL:|of| CONTEXT)
                                       (IL:|fetch| SELECT-STRING IL:|of| PENDING-SELECTION))
                                (UPDATE CONTEXT)
                                (IL:TTY.PROCESS (IL:WINDOWPROP WINDOW 'IL:PROCESS)))
                               (T
```

;; copy or move -- figure out whether it's going into an SEdit, or to an unknown sink (in which case we print ;; it)

```
                                  (WHEN DESTINATION-CONTEXT        ; it's going to an SEdit. prepare it
                                      (IL:\\CARET.DOWN (IL:|fetch| DISPLAY-WINDOW IL:|of| DESTINATION-CONTEXT))
                                      (SELECTION-DOWN DESTINATION-CONTEXT)
                                      (CLOSE-OPEN-NODE DESTINATION-CONTEXT)
                                      (IL:SETQ DESTINATION-POINT (IL:|fetch| CARET-POINT IL:|of| DESTINATION-CONTEXT))
)
                                  (COPY-SELECTION PENDING-SELECTION CONTEXT DESTINATION-CONTEXT
                                         DESTINATION-POINT (EQ PENDING-SHIFT 'MOVE))
                                  (WHEN (IL:NEQ CONTEXT DESTINATION-CONTEXT)
                                      (COND
                                         ((EQ PENDING-SHIFT 'MOVE)
                                          (UPDATE CONTEXT))
                                         ((IL:OBTAIN.MONITORLOCK (IL:|fetch| CONTEXT-LOCK IL:|of| CONTEXT)
                                                 T)
```

;; for Copy select, only display the selection if this is a non-busy sedit

```
                                          (SELECTION-UP CONTEXT)
                                          (IL:RELEASE.MONITORLOCK (IL:|fetch| CONTEXT-LOCK IL:|of| CONTEXT)))))
                                  (WHEN DESTINATION-CONTEXT
```

;; just wake up the destination and let it update itself.

```
                                        (AWAKE-COMMAND-PROCESS DESTINATION-CONTEXT)))))))
            (T ;; just setting the current selection, and maybe the caret.  it is all displayed from when it was pending, so mark it as displayed now
               (IL:|replace| SELECTION-DISPLAYED? IL:|of| CONTEXT IL:|with| T)
               ;; and make it the main selection and point
               (SMASH-USING EDIT-SELECTION SELECTION PENDING-SELECTION)
               (IL:|replace| LAST-MOUSE-X IL:|of| CONTEXT IL:|with| PENDING-LAST-X)
               (IL:|replace| LAST-MOUSE-Y IL:|of| CONTEXT IL:|with| PENDING-LAST-Y)
               (IL:|replace| LAST-MOUSE-TYPE IL:|of| CONTEXT IL:|with| PENDING-TYPE)
               (WHEN (IL:|fetch| PENDING-DELETE? IL:|of| PENDING-SELECTION)
                   (IL:|replace| POINT-NODE IL:|of| PENDING-CARET IL:|with| SELECTION)
                   (IL:|replace| POINT-TYPE IL:|of| PENDING-CARET IL:|with| (IL:|fetch| SELECT-TYPE IL:|of| PENDING-SELECTION)
                     ))
               (SMASH-USING EDIT-POINT (IL:|fetch| CARET-POINT IL:|of| CONTEXT)
                     PENDING-CARET)
               (SHOW-CARET CONTEXT))))))
```

(FIND-LINE-START

```
(IL:LAMBDA (Y CONTEXT)                                  ; Edited 17-Nov-87 11:22 by DCB
```

;;; find the line including a given y coordinate. very dumb -- we just linear search through them -- but does the job

```
(IL:BIND (LINE IL:_ (IL:FETCH LINEAR-FORM IL:OF (IL:FETCH ROOT IL:OF CONTEXT)))
       NEXT-LINE IL:FIRST (WHEN (OR (IL:ILEQ Y (IL:FETCH IL:BOTTOM IL:OF (IL:WINDOWPROP (IL:FETCH
                                                                                  DISPLAY-WINDOW
                                                                          IL:OF CONTEXT)
                                                                  'IL:EXTENT)))
                                    (IL:IGREATERP Y 0))       ; above or below the structure
                                (RETURN NIL))
   IL:DO (IF (AND (IL:SETQ NEXT-LINE (IL:FETCH NEXT-LINE IL:OF (CAR LINE)))
                  (IL:IGEQ (IL:FETCH YCOORD IL:OF (CAR NEXT-LINE))
                           Y))
             (IL:SETQ LINE NEXT-LINE)
             (RETURN LINE)))))
```

(FIND-NODE

```
(IL:LAMBDA (X LINEAR-POINTER CONTEXT)                   ; Edited 17-Nov-87 11:22 by DCB
```

;;; sort of a dubious name. we're actually trying to find the linear item on this line which corresponds to the given x position. linear.pointer is the line. as
;;; an added bonus, set the \X field of context to the x coordinate of this linear item. this is a hack; we really want to return multiple values, but there's no
;;; clean way to do that in interlisp

```
(PROG (LINEAR-ITEM)
      (WHEN (IL:ILESSP X 0)
          ;; to the left of the whole structure -- nothing there! (i don't think this should ever happen)
          (RETURN NIL))
      (IL:SETQ LINEAR-ITEM (CAR LINEAR-POINTER))
      (WHEN (IL:IGEQ X (IL:FETCH LINE-LENGTH IL:OF LINEAR-ITEM))
          ;; past the right edge of this line;  say we're before the next line
          (IL:REPLACE \\X IL:OF CONTEXT IL:WITH 1)
          (RETURN (IL:FETCH NEXT-LINE IL:OF LINEAR-ITEM)))
      (IL:BIND (CURRENT-X IL:_ 0)
            (NEXTX IL:_ (IL:FETCH INDENT IL:OF LINEAR-ITEM)) IL:WHILE (IL:ILEQ NEXTX X)
         IL:DO (IL:SETQ CURRENT-X NEXTX)
               (IL:SETQ LINEAR-POINTER (NEXT-LINEAR-ITEM (CDR LINEAR-POINTER)))
               (IL:SETQ NEXTX (IL:IPLUS NEXTX (LINEAR-ITEM-WIDTH (CAR LINEAR-POINTER))))
         IL:FINALLY (IL:REPLACE \\X IL:OF CONTEXT IL:WITH CURRENT-X))
      (RETURN LINEAR-POINTER))))
```

(GET-DESTINATION-CONTEXT

```
(IL:LAMBDA NIL                                          ; Edited  7-Jul-87 13:03 by DCB
```

;;; used under shift selections. if the destination is an SEdit, return its context, otherwise NIL. It is considered a valid (ready for shift selection) SEdit if
;;; the process is waiting under getkey

```
(LET ((DESTINATION (IL:PROCESSPROP (IL:TTY.PROCESS)
                          'IL:WINDOW)))
     (AND DESTINATION (IL:SETQ DESTINATION (IL:WINDOWPROP DESTINATION 'EDIT-CONTEXT))
          (IL:PROCESS.EVAL (IL:TTY.PROCESS)
                 '(IL:STKPOS 'GETKEY)
                 T)
          DESTINATION))))
```

(GRAY

```
(IL:LAMBDA (WINDOW)                                     ; Edited 17-Nov-87 11:23 by DCB
```

;;; due to a misfeature of the window system, we have to adjust the gray texture depending on how much the window's been scrolled. bleah

```
    ;; DEdit's SHADEFIXER handles the more general case
```

```
    (IF (EQ (EVENP (IL:DSPXOFFSET NIL WINDOW))
            (EVENP (IL:DSPYOFFSET NIL WINDOW)))
        23130
        42405)))
```

(GROW-CLICK?

```
  (IL:LAMBDA (CONTEXT POINT-TYPE WINDOW)                    ; Edited  7-Jul-87 13:03 by DCB
```

;; the left or middle mouse button is down. decide if this is part of a multi-click, i.e. the mouse stays in the same position as the previous click. if ;; so, we just grow the selection. return T if that's what happened

```
    (WHEN (AND (COND
                   ((IL:FETCH SELECT-NODE IL:OF PENDING-SELECTION) ; you can't grow a selection if you've already extended it
                    (NOT (IL:FETCH SELECT-END IL:OF PENDING-SELECTION)))
                   (T (AND (NOT PENDING-SHIFT)
                           (IL:FETCH SELECT-NODE IL:OF (IL:FETCH SELECTION IL:OF CONTEXT))
                           (NOT (IL:FETCH SELECT-END IL:OF (IL:FETCH SELECTION IL:OF CONTEXT))))))
               (EQ PENDING-TYPE POINT-TYPE)
               (IL:ILEQ (ABS (IL:IDIFFERENCE (IL:LASTMOUSEX WINDOW)
                                             PENDING-LAST-X))
                        2)
               (IL:ILEQ (ABS (IL:IDIFFERENCE (IL:LASTMOUSEY WINDOW)
                                             PENDING-LAST-Y))
                        2))
```

;; it looks like we've got a grow click. display the grown selection, and wait until the mouse button goes up

```
        (COND
           ((IL:FETCH SELECT-NODE IL:OF PENDING-SELECTION)        ; turn off the previous selection
            (DISPLAY-SELECTION PENDING-SELECTION WINDOW PENDING-SHIFT))
           (T (SMASH-USING EDIT-SELECTION PENDING-SELECTION (IL:FETCH SELECTION IL:OF CONTEXT))))
        (GROW-SELECTION PENDING-SELECTION CONTEXT)
        (WHEN (AND (IL:FETCH SELECT-NODE IL:OF PENDING-SELECTION)
                   (NULL (IL:FETCH SELECT-START-X IL:OF PENDING-SELECTION)))
              (COMPUTE-SELECTION-POSITION PENDING-SELECTION CONTEXT))
        (DISPLAY-SELECTION PENDING-SELECTION WINDOW PENDING-SHIFT)
        (SET-POINT-NOWHERE PENDING-CARET)
        (IL:DO ;; keep watching for new modifier keys, until the mouse buttons come up *or* the cursor is moved, which cancels the grow
               (CHECK-SELECTION-SHIFT CONTEXT)
               (IL:BLOCK) IL:REPEATUNTIL (OR (IL:MOUSESTATE (OR IL:UP IL:RIGHT))
                                             (IL:IGREATERP (ABS (IL:IDIFFERENCE (IL:LASTMOUSEX WINDOW)
                                                                                PENDING-LAST-X))
                                                           2)
                                             (IL:IGREATERP (ABS (IL:IDIFFERENCE (IL:LASTMOUSEY WINDOW)
                                                                                PENDING-LAST-Y))
                                                           2)))
        (IL:MOUSESTATE (OR IL:UP IL:RIGHT)))))
```

(GROW-SELECTION

```
  (IL:LAMBDA (SELECTION CONTEXT)                            ; Edited 17-Nov-87 11:23 by DCB
```

;;; compute the new selection which results from growing this one

```
    (FUNCALL (IL:FETCH GROW-SELECTION IL:OF (IL:FETCH NODE-TYPE IL:OF (IL:FETCH SELECT-NODE IL:OF SELECTION)))
           SELECTION CONTEXT (IL:FETCH SELECT-NODE IL:OF SELECTION))))
```

(GROW-SELECTION-DEFAULT

```
  (IL:LAMBDA (SELECTION CONTEXT NODE)                       ; Edited 17-Nov-87 11:23 by DCB
```

;;; a default method for GrowSelection. if we're not the top node in the tree (i.e. our super isn't the root) then select our super

```
    (WHEN (IL:FETCH SUPER-NODE IL:OF (IL:FETCH SUPER-NODE IL:OF NODE))
          (PUNT-SET-SELECTION SELECTION CONTEXT NODE))))
```

(HIGHLIGHT-SELECTION

```
  (IL:LAMBDA (SELECTION WINDOW SHADE)                       ; Edited 17-Nov-87 11:23 by DCB
```

;;; highlight this selection. draw.highlight does all the work, once we've figured out the bounds

```
    (OUTLINE-SELECTION SELECTION WINDOW SHADE (IL:FUNCTION DRAW-HIGHLIGHT))))
```

(ICON-COPYFN

```
  (IL:LAMBDA (IL:WINDOW)                                    ; Edited  8-Jan-88 09:00 by DCB
```

;;; BKSYSBUFs the title of the SEdit window (as a structure if it is one)

```
    (LET ((NAME (IL:LISTGET (IL:WINDOWPROP (IL:WINDOWPROP IL:WINDOW 'IL:ICONFOR)
                                           'TITLE-INFO)
                            :|name|)))
         (IF NAME
             (IL:BKSYSBUF NAME T)
             (IL:BKSYSBUF " " NIL)))))
```

(LESS-PROMPT-WINDOW

```
(IL:LAMBDA (REGION FONT)                                    ; Edited  7-Jul-87 13:03 by DCB
  (IL:CREATEREGION (IL:FETCH (IL:REGION IL:LEFT) IL:OF REGION)
         (IL:FETCH (IL:REGION IL:BOTTOM) IL:OF REGION)
         (IL:FETCH (IL:REGION IL:WIDTH) IL:OF REGION)
         (IL:IDIFFERENCE (IL:FETCH (IL:REGION IL:HEIGHT) IL:OF REGION)
                (IL:HEIGHTIFWINDOW (IL:FONTPROP FONT 'IL:HEIGHT))))))
```

(NORMALIZE-SELECTION

```
(IL:LAMBDA (CONTEXT)                                        ; Edited  7-Jul-87 13:03 by DCB
```

;;; if the current selection isn't visible in the window, scroll until it is. we only worry about vertical position; this could be extended to handle horizontal ;;; scrolling too, should there prove any need. since we're usually getting called after just setting the selection to be normalized, we have to compute the ;;; position first, in order to know how to center it.

```
    (LET ((SELECTION (IL:FETCH SELECTION IL:OF CONTEXT))
          (REGION (IL:DSPCLIPPINGREGION NIL (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)))
          FIRST-LINE)
         (COMPUTE-SELECTION-POSITION SELECTION CONTEXT)
         (IL:SETQ FIRST-LINE (IL:FETCH SELECT-START-LINE IL:OF SELECTION))
         (WHEN (OR (IL:ILESSP (IL:FETCH NEXT-LINE-Y IL:OF FIRST-LINE)
                              (IL:FETCH (IL:REGION IL:BOTTOM) IL:OF REGION))
                   (IL:IGREATERP (IL:FETCH YCOORD IL:OF FIRST-LINE)
                              (IL:FETCH (IL:REGION IL:TOP) IL:OF REGION)))
```

;; the selection isn't completely visible. scroll so that the top of it is 1/3 of the way from the top of the window. it might still not be ;; completely visible, but it's good enough

```
               (IL:SCROLLW (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                      0
                      (IL:IDIFFERENCE (IL:FETCH (IL:REGION IL:TOP) IL:OF REGION)
                             (IL:IMIN 0 (IL:IPLUS (IL:FETCH YCOORD IL:OF FIRST-LINE)
                                                  (IL:IQUOTIENT (IL:FETCH (IL:REGION IL:HEIGHT) IL:OF REGION)
                                                         3)))))))))
```

(OUTLINE-SELECTION

```
(IL:LAMBDA (SELECTION WINDOW SHADE FN)                      ; Edited 17-Nov-87 11:23 by DCB
```

;;; highlight this selection. draw.outline does all the work, once we've figured out the bounds. we also share this code with highlight.selection, via a ;;; functional parameter

```
    (IL:BIND (MINX IL:_ (IL:FETCH SELECT-START-X IL:OF SELECTION))
          (MAXX IL:_ (IL:FETCH SELECT-END-X IL:OF SELECTION))
          (LINE IL:_ (IL:FETCH SELECT-START-LINE IL:OF SELECTION))
          (ENDLINE IL:_ (IL:FETCH SELECT-END-LINE IL:OF SELECTION)) IL:WHILE (IL:NEQ LINE ENDLINE)
       IL:DO (IL:SETQ MAXX (IL:IMAX MAXX (IL:FETCH LINE-LENGTH IL:OF LINE)))
          (IL:SETQ LINE (CAR (IL:FETCH NEXT-LINE IL:OF LINE)))
          (IL:SETQ MINX (IL:IMIN MINX (IL:FETCH INDENT IL:OF LINE)))
       IL:FINALLY (FUNCALL (OR FN (IL:FUNCTION DRAW-OUTLINE))
                           MINX
                           (IL:FETCH SELECT-START-X IL:OF SELECTION)
                           (IL:FETCH SELECT-END-X IL:OF SELECTION)
                           (IL:IDIFFERENCE MAXX MINX)
                           (IL:FETCH YCOORD IL:OF (IL:FETCH SELECT-START-LINE IL:OF SELECTION))
                           (IL:FETCH LINE-HEIGHT IL:OF (IL:FETCH SELECT-START-LINE IL:OF SELECTION))
                           (IL:FETCH YCOORD IL:OF ENDLINE)
                           (IL:FETCH LINE-HEIGHT IL:OF ENDLINE)
                           WINDOW SHADE))))
```

(PENDING-DELETE

```
(IL:LAMBDA (POINT SELECTION)                                ; Edited  7-Jul-87 13:03 by DCB
  (WHEN (IL:FETCH SELECT-NODE IL:OF SELECTION)
      (IL:REPLACE PENDING-DELETE? IL:OF SELECTION IL:WITH T)
      (IL:REPLACE POINT-NODE IL:OF POINT IL:WITH SELECTION)
      (IL:REPLACE POINT-TYPE IL:OF POINT IL:WITH (IL:FETCH SELECT-TYPE IL:OF SELECTION)))))
```

(PLACE-CARET-AND-SELECTION

```
(IL:LAMBDA (CARET SELECTION CONTEXT X Y TYPE)                ; Edited 17-Nov-87 11:24 by DCB
```

;;; compute the new location of the caret and current selection, given the coordintes of the mouse and the type of selection being made

```
    (LET (LINE LINEAR NODE INDEX OFFSET)
         (COND
            ((AND (IL:INSIDEP (IL:DSPCLIPPINGREGION NIL (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT))
                          X Y)
                  (IL:SETQ LINE (FIND-LINE-START Y CONTEXT))
                  (IL:SETQ LINEAR (FIND-NODE X LINE CONTEXT)))
```

;; we've found the linear item they're pointing at. figure out what node it belongs to, what its index in the linear form is, and how far into ;; the item the position is

```
             (IL:SETQ NODE (IL:FETCH DESTINATION IL:OF (CDR (LAST LINEAR))))
             (IL:SETQ INDEX (IL:FOR I IL:FROM 1 IL:AS (X IL:_ (IL:FETCH LINEAR-FORM IL:OF NODE))
                               IL:BY (CDR X) IL:THEREIS (EQ X LINEAR)))
```

```
            (IL:SETQ OFFSET (IF (IL:TYPE? LINE-START (CAR LINEAR))
                                (IF (EQ 0 (IL:FETCH \\X IL:OF CONTEXT))
                                    1
                                    -1)
                                (IL:IDIFFERENCE X (IL:FETCH \\X IL:OF CONTEXT))))
```

;; call the appropriate methods to place the point and selection

```
            (WHEN CARET
                (SET-POINT CARET CONTEXT NODE INDEX OFFSET (CAR LINEAR)
                       TYPE T))
            (SET-SELECTION SELECTION CONTEXT NODE INDEX OFFSET (CAR LINEAR)
                    TYPE)
            (WHEN (AND (IL:FETCH SELECT-NODE IL:OF SELECTION)
                       (NULL (IL:FETCH SELECT-START-X IL:OF SELECTION)))
                  (COMPUTE-SELECTION-POSITION SELECTION CONTEXT)))
           (T
```

;; the mouse isn't pointing at anything -- cancel the point and selection

```
              (WHEN CARET (SET-POINT-NOWHERE CARET))
              (SET-SELECTION-NOWHERE SELECTION))))))
```

(PUNT-SET-POINT

```
(IL:LAMBDA (POINT CONTEXT NODE WHICH-END COMPUTE-LOCATION?)     ; Edited 17-Nov-87 11:24 by DCB
```

;; there's no place to put the point in this node; try letting the supernode put it immediately before or after this node -- before if which.end is NIL,
;; after if it's T

```
    (SET-POINT POINT CONTEXT (IL:FETCH SUPER-NODE IL:OF NODE)
           (IL:FETCH SUB-NODE-INDEX IL:OF NODE)
           WHICH-END NODE 'STRUCTURE COMPUTE-LOCATION?)))
```

(PUNT-SET-SELECTION

```
(IL:LAMBDA (SELECTION CONTEXT NODE)                             ; Edited 17-Nov-87 11:24 by DCB
```

;;; this node can't handle the selection; ask its supernode to try

```
    (SET-SELECTION SELECTION CONTEXT (IL:FETCH SUPER-NODE IL:OF NODE)
           (IL:FOR I IL:FROM 1 IL:AS (X IL:_ (IL:FETCH LINEAR-FORM IL:OF (IL:FETCH SUPER-NODE IL:OF NODE)))
              IL:BY (CDR X) IL:THEREIS (EQ X (IL:FETCH LINEAR-THREAD IL:OF NODE)))
           NIL NODE 'STRUCTURE)))
```

(REPAINTFN

```
(IL:LAMBDA (WINDOW REGION)                                      ; Edited  7-Jul-87 13:03 by DCB
```

;; called by the window system when it needs some or all of the window to be repainted (based on region)

```
    (LET ((CONTEXT (IL:WINDOWPROP WINDOW 'EDIT-CONTEXT)))
         (WHEN CONTEXT
             (WITH-PROFILE (IL:FETCH PROFILE IL:OF CONTEXT)
                    (LET (START LINE)
                         (WHEN (IL:SETQ START (FIND-LINE-START (IL:FETCH (IL:REGION IL:TOP) IL:OF REGION)
                                                               CONTEXT))
                             (IL:SETQ LINE (CAR START))
```

;; here we have to lie about the selection. it may have been displayed, but now the region has been cleared, so
;; that part of the selection is no longer on the screen. setting the flag NIL will force it to be redisplayed on the
;; way out.

```
                             (IL:REPLACE SELECTION-DISPLAYED? IL:OF CONTEXT IL:WITH NIL)
                             (REPAINT CONTEXT (IL:FETCH INDENT IL:OF LINE)
                                    (IL:FETCH BASE-LINE-Y IL:OF LINE)
                                    (CDR START)
                                    (IL:FETCH (IL:REGION IL:BOTTOM) IL:OF REGION))
                             (WHEN (EQ SELECTION-PENDING? CONTEXT)
```

;; they're in the process of making a selection in this window -- probably scrolling to extend the selection
; (fix.caret.position)

```
                                 (DISPLAY-SELECTION PENDING-SELECTION WINDOW PENDING-SHIFT))
```

;; now that we're done, try to bring back the main selection.

```
                             (SELECTION-UP CONTEXT)))))))
```

(RESHAPEFN

```
(IL:LAMBDA (WINDOW OLD-IMAGE OLD-IMAGE-REGION OLD-SCREEN-REGION)
                                                                ; Edited  7-Jul-87 13:04 by DCB
```

;; called by the window system when the window's size changes. if the width is exactly the same we'll just reuse as much of the image as possible
;; and repaint the rest. if the width has changed, we'll have to completely reformat

```
    (LET* ((CONTEXT (IL:WINDOWPROP WINDOW 'EDIT-CONTEXT))
           (NEW-REGION (IL:DSPCLIPPINGREGION NIL WINDOW))
           (OLD-BOTTOM (IL:FETCH (IL:REGION IL:BOTTOM) IL:OF NEW-REGION)))
          (IL:WYOFFSET (IL:IDIFFERENCE (IL:FETCH (IL:REGION IL:HEIGHT) IL:OF NEW-REGION)
                                       (IL:FETCH (IL:REGION IL:HEIGHT) IL:OF OLD-IMAGE-REGION))
                       WINDOW)
          (COMPUTE-COMMENT-COLUMN CONTEXT WINDOW)
          (IL:WITH.MONITOR (IL:FETCH CONTEXT-LOCK IL:OF CONTEXT)
              (COND
                 ((EQ (IL:FETCH (IL:REGION IL:WIDTH) IL:OF OLD-IMAGE-REGION)
```

```
                    (IL:FETCH (IL:REGION IL:WIDTH) IL:OF NEW-REGION))
                                                                    ; reuse the old bits
                (IL:BITBLT OLD-IMAGE (IL:FETCH (IL:REGION IL:LEFT) IL:OF OLD-IMAGE-REGION)
                       (IL:FETCH (IL:REGION IL:BOTTOM) IL:OF OLD-IMAGE-REGION)
                       WINDOW
                       (IL:FETCH (IL:REGION IL:LEFT) IL:OF NEW-REGION)
                       OLD-BOTTOM
                       (IL:FETCH (IL:REGION IL:WIDTH) IL:OF OLD-IMAGE-REGION)
                       (IL:FETCH (IL:REGION IL:HEIGHT) IL:OF OLD-IMAGE-REGION))
                (WHEN (IL:IGREATERP (IL:FETCH (IL:REGION IL:HEIGHT) IL:OF NEW-REGION)
                              (IL:FETCH (IL:REGION IL:HEIGHT) IL:OF OLD-IMAGE-REGION))
                    ;; if the new one is smaller, we're done.  otherwise we have to repaint the extra space
                    (LET ((BLANK-REGION (IL:CREATE IL:REGION IL:USING NEW-REGION IL:HEIGHT IL:_
                                                   (IL:IDIFFERENCE (IL:FETCH (IL:REGION
                                                                                 IL:HEIGHT)
                                                                             IL:OF NEW-REGION)
                                                                   (IL:FETCH (IL:REGION IL:HEIGHT)
                                                                             IL:OF OLD-IMAGE-REGION)))))
                          (IL:RESETLST
                              (IL:RESETSAVE NIL (LIST 'IL:DSPCLIPPINGREGION (IL:DSPCLIPPINGREGION BLANK-REGION
                                                                                                  WINDOW)
                                                    WINDOW))
                              ;; clip to area to repaint, and make sure clipping region gets reset on the way out.
                              (IL:REPAINTFN WINDOW BLANK-REGION)))))
                (T ;; the new window is a different width.  reformat and repaint from scratch.  we also cancel any horizontal scrolling
                   (WITH-PROFILE (IL:FETCH PROFILE IL:OF CONTEXT)
                          (IL:WXOFFSET (IL:FETCH (IL:REGION IL:LEFT) IL:OF NEW-REGION)
                                 WINDOW)
                          ;; atom.change.relinearize is just a convenient way to close up sedit structure and relinearize from scratch.
                          (ATOM-CHANGE-RELINEARIZE CONTEXT)))))))
```

(SCAN-FOR-BOUNDS

```
  (IL:LAMBDA (START END LINE INITIALIZE)                    ; Edited 11-Apr-88 15:26 by woz

;;; we have to recompute the ascent and descent of this line.  scan the linear form from start to end (or the next line start, which ever comes first) and
;;; compute the maximum ascent and descent.  we also fix up the first and last line fields of any nodes we notice, and compute and return the width of
;;; the section of linear form we examine

    (IL:|bind| ITEM ITEM-NODE (LINE-START IL:_ (CAR LINE))
          MAX-ASCENT MAX-DESCENT (X IL:_ 0) IL:|first| (COND
                                                         (INITIALIZE (IL:SETQ MAX-ASCENT 0)
                                                                (IL:SETQ MAX-DESCENT 0))
                                                         (T (IL:SETQ MAX-ASCENT (IL:|fetch| LINE-ASCENT IL:|of|
                                                                                                    LINE-START
                                                                                             ))
                                                            (IL:SETQ MAX-DESCENT (IL:|fetch| LINE-DESCENT
                                                                                         IL:|of| LINE-START))))
       IL:|do| (WHEN (EQ START END)
                   (WHEN (NULL START)
                         (IL:|replace| NEXT-LINE IL:|of| LINE-START IL:|with| NIL))
                   (GO IL:$$OUT))
               (COND
                  ((IL:LISTP START)
                   (IL:SETQ ITEM (CAR START))
                   (COND
                      ((IL:|type?| WEAK-LINK ITEM)
                       (SETQ ITEM-NODE (IL:|fetch| DESTINATION IL:|of| ITEM))
                       (IL:|replace| FIRST-LINE IL:|of| ITEM-NODE IL:|with| LINE-START)
                       (IL:SETQ START (IL:|fetch| LINEAR-FORM IL:|of| ITEM-NODE)))
                      ((IL:|type?| LINE-START ITEM)
                       (IL:|replace| PREV-LINE IL:|of| ITEM IL:|with| LINE)
                       (IL:|replace| NEXT-LINE IL:|of| LINE-START IL:|with| START)
                       (GO IL:$$OUT))
                      (T (COND
                            ((IL:FIXP ITEM)
                             (IL:SETQ X (IL:IPLUS X ITEM)))
                            ((IL:|type?| STRING-ITEM ITEM)
                             (IL:SETQ X (IL:IPLUS X (IL:|fetch| WIDTH IL:|of| ITEM)))
                             (IL:SETQ ITEM (IL:|fetch| FONT IL:|of| ITEM))
                             (IL:SETQ MAX-ASCENT (IL:IMAX MAX-ASCENT (IL:FONTPROP ITEM 'IL:ASCENT)))
                             (IL:SETQ MAX-DESCENT (IL:IMAX MAX-DESCENT (IL:FONTPROP ITEM 'IL:DESCENT))))
                            (T (IL:SETQ MAX-ASCENT (IL:IMAX MAX-ASCENT (IL:IDIFFERENCE (IL:BITMAPHEIGHT
                                                                                           (CDR ITEM))
                                                                                       (CAR ITEM))))
                               (IL:SETQ MAX-DESCENT (IL:IMAX MAX-DESCENT (IL:IMINUS (CAR ITEM))))
                               (IL:SETQ X (IL:IPLUS X (IL:BITMAPWIDTH (CDR ITEM))))))
                         (IL:SETQ START (CDR START)))))
                  (T (IL:SETQ START (IL:|fetch| DESTINATION IL:|of| START))
                                                                    ; used to replace LastLineLinear of start with line
                     (IL:|replace| LAST-LINE IL:|of| START IL:|with| LINE-START)
                     (IL:SETQ START (CDR (IL:|fetch| LINEAR-THREAD IL:|of| START)))))
       IL:|finally| (IL:|replace| LINE-ASCENT IL:|of| LINE-START IL:|with| MAX-ASCENT)
```

```
            (IL:|replace| LINE-DESCENT IL:|of| LINE-START IL:|with| MAX-DESCENT)
            (WHEN (IL:|type?| WEAK-LINK START)                    ; used to replace LastLineLinear of (fetch Destination of start)
                                                                  ; with line
                (IL:|replace| LAST-LINE IL:|of| (IL:|fetch| DESTINATION IL:|of| START) IL:|with| (CAR LINE)))
            (RETURN X))))
```

(SELECT-NODE

```
(IL:LAMBDA (CONTEXT NODE SET-POINT? WHERE)                        ; Edited 3-Dec-87 12:15 by DCB
  (SET-SELECTION-ME (IL:FETCH SELECTION IL:OF CONTEXT)
        CONTEXT NODE)
  (IL:REPLACE PENDING-DELETE? IL:OF (IL:FETCH SELECTION IL:OF CONTEXT) IL:WITH NIL)
  (WHEN SET-POINT?
      (SET-POINT (IL:FETCH CARET-POINT IL:OF CONTEXT)
          CONTEXT NODE NIL WHERE NIL 'STRUCTURE T))))
```

(SELECT-SEGMENT

```
(IL:LAMBDA (SELECTION CONTEXT NODE SUBNODE INDEX OFFSET ITEM)  ; Edited 17-Nov-87 11:24 by DCB
```

;;; apply the appropriate SelectSegment method to set this selection

```
  (IL:REPLACE DELETE-OK? IL:OF SELECTION IL:WITH T)
  (IL:REPLACE PENDING-DELETE? IL:OF SELECTION IL:WITH T)
  (FUNCALL (IL:FETCH SELECT-SEGMENT IL:OF (IL:FETCH NODE-TYPE IL:OF NODE))
        SELECTION CONTEXT NODE SUBNODE INDEX OFFSET ITEM)))
```

(SELECT-SEGMENT-DEFAULT

```
(IL:LAMBDA (SELECTION CONTEXT NODE SUBNODE INDEX OFFSET ITEM)  ; Edited 11-Apr-88 15:26 by woz
```

;;; a default SelectSegment method for aggregate types. selects the sequence of subnodes bounded by the selected items

```
  (LET (START END)
       (COND
          (SUBNODE (IL:SETQ START (IL:SETQ END (IL:|fetch| SUB-NODE-INDEX IL:|of| SUBNODE))))
          (T (IL:SETQ START (IL:|fetch| SELECT-START IL:|of| SELECTION))
             (IL:SETQ END (OR (IL:|fetch| SELECT-END IL:|of| SELECTION)
                              START))))
       (COND
          ((NULL INDEX)
           (IL:SETQ START (IL:IMIN START (IL:|fetch| SELECT-START IL:|of| SELECTION)))
           (IL:SETQ END (IL:IMAX END (IL:|fetch| SELECT-END IL:|of| SELECTION))))
          ((IL:|type?| EDIT-NODE INDEX)
           (COND
              ((IL:ILESSP (IL:SETQ INDEX (IL:|fetch| SUB-NODE-INDEX IL:|of| INDEX))
                      START)
               (IL:SETQ START INDEX))
              ((IL:IGREATERP INDEX END)
               (IL:SETQ END INDEX))))
          (T (IL:|for| LINEAR-ITEM IL:|in| (IL:|fetch| LINEAR-FORM IL:|of| NODE) IL:|as| LINEAR-INDEX IL:|from| 1
                IL:|bind| LAST-SUBNODE-INDEX TAKE-NEXT LINEAR-ITEM-NODE
                IL:|do| (WHEN (IL:|type?| WEAK-LINK LINEAR-ITEM)
                              (SETQ LINEAR-ITEM-NODE (IL:|fetch| DESTINATION IL:|of| LINEAR-ITEM))
                              (COND
                                 (TAKE-NEXT (RETURN (IL:SETQ START (IL:IMIN START (IL:|fetch| SUB-NODE-INDEX
                                                                                      IL:|of| LINEAR-ITEM-NODE)))))
                                 (T (IL:SETQ LAST-SUBNODE-INDEX (IL:|fetch| SUB-NODE-INDEX IL:|of| LINEAR-ITEM-NODE))
                                    (WHEN (EQ LINEAR-INDEX INDEX)
                                        (COND
                                           ((IL:ILESSP LAST-SUBNODE-INDEX START)
                                            (IL:SETQ START LAST-SUBNODE-INDEX))
                                           ((IL:IGREATERP LAST-SUBNODE-INDEX END)
                                            (IL:SETQ END LAST-SUBNODE-INDEX)))
                                        (RETURN)))))
                          (WHEN (EQ LINEAR-INDEX INDEX)
                              (IF (AND LAST-SUBNODE-INDEX (IL:IGEQ LAST-SUBNODE-INDEX START))
                                  (RETURN (IL:SETQ END (IL:IMAX END LAST-SUBNODE-INDEX)))
                                  (IL:SETQ TAKE-NEXT T))))))
       (IL:|replace| SELECT-NODE IL:|of| SELECTION IL:|with| NODE)
       (IL:|replace| SELECT-START IL:|of| SELECTION IL:|with| START)
       (IL:|replace| SELECT-END IL:|of| SELECTION IL:|with| END)
       (IL:|replace| SELECT-START-X IL:|of| SELECTION IL:|with| NIL)
       (IL:|replace| SELECT-TYPE IL:|of| SELECTION IL:|with| 'STRUCTURE))))
```

(SELECTION-DOWN

```
(IL:LAMBDA (CONTEXT)                                              ; Edited 7-Jul-87 13:04 by DCB
```

;;; turn off the display of the current selection -- we're going to change the window. displaly.se

```
  (WHEN (IL:FETCH SELECTION-DISPLAYED? IL:OF CONTEXT)
      (DISPLAY-SELECTION (IL:FETCH SELECTION IL:OF CONTEXT)
            (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT))
      (IL:REPLACE SELECTION-DISPLAYED? IL:OF CONTEXT IL:WITH NIL))))
```

(SELECTION-UP

```
(IL:LAMBDA (CONTEXT)                                    ; Edited  7-Jul-87 13:04 by DCB
```

;;; make sure the selection is displayed. if it's not, and displaying it works, then mark it as displayed.

```
    (WHEN (AND (NOT (IL:FETCH SELECTION-DISPLAYED? IL:OF CONTEXT))
               (DISPLAY-SELECTION (IL:FETCH SELECTION IL:OF CONTEXT)
                                  (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)))
          (IL:REPLACE SELECTION-DISPLAYED? IL:OF CONTEXT IL:WITH T))))
```

(SET-POINT

```
(IL:LAMBDA (POINT CONTEXT NODE INDEX OFFSET ITEM TYPE COMPUTE-LOCATION?)
                                                        ; Edited  7-Jul-87 13:04 by DCB
```

;;; apply the appropriate SetPoint method to set this point. these methods must be able to handle 3 cases:

;;; case 1: index is index into linear form of cursor, offset is offset into that item, item is the item

;;; case 2: (set point at beginning or end of this node) : index is NIL, offset is NIL for beginning, T for end

;;; case 3: (set point before or after subnode) : index is subnode index, offset is before/after, item is subnode

```
    (FUNCALL (IL:FETCH SET-POINT IL:OF (IL:FETCH NODE-TYPE IL:OF NODE))
           POINT CONTEXT NODE INDEX OFFSET ITEM TYPE COMPUTE-LOCATION?)))
```

(SET-POINT-NOWHERE

```
(IL:LAMBDA (POINT)                                      ; Edited 17-Nov-87 11:25 by DCB
```

;;; a SetPoint method for types that have nowhere to insert

```
    (IL:REPLACE POINT-NODE IL:OF POINT IL:WITH NIL)
    (IL:REPLACE POINT-TYPE IL:OF POINT IL:WITH NIL)))
```

(SET-POINT-UNKNOWN

```
(IL:LAMBDA (POINT CONTEXT NODE INDEX OFFSET ITEM TYPE COMPUTE-LOCATION?)
                                                        ; Edited 17-Nov-87 11:25 by DCB
```

;;; the SetPoint method for type unknown, and anyone else doesn't allow insertions but whose super might. ask the super to except input before or after
;;; this node, based on which is closer. note that the calculation for which is closer assumes that the node is displayed inline, so this method won't work
;;; for anyone that doesn't

```
    (PUNT-SET-POINT POINT CONTEXT NODE (IF INDEX
                                           (IL:IGEQ OFFSET (IL:HALF (IL:FETCH INLINE-WIDTH IL:OF NODE)))
                                           OFFSET)
           COMPUTE-LOCATION?)))
```

(SET-SELECTION

```
(IL:LAMBDA (SELECTION CONTEXT NODE INDEX OFFSET ITEM TYPE)      ; Edited 17-Nov-87 11:25 by DCB
```

;;; apply the appropriate SetSelection method to set this selection

```
    (IL:REPLACE DELETE-OK? IL:OF SELECTION IL:WITH T)
    (IL:REPLACE PENDING-DELETE? IL:OF SELECTION IL:WITH NIL)
    (FUNCALL (IL:FETCH SET-SELECTION IL:OF (IL:FETCH NODE-TYPE IL:OF NODE))
           SELECTION CONTEXT NODE INDEX OFFSET ITEM TYPE)))
```

(SET-SELECTION-ME

```
(IL:LAMBDA (SELECTION CONTEXT NODE)                     ; Edited 17-Nov-87 11:26 by DCB
```

;;; set the current selection to be this node

```
    (IL:|replace| SELECT-NODE IL:|of| SELECTION IL:|with| NODE)
    (IL:|replace| SELECT-START IL:|of| SELECTION IL:|with| NIL)
    (IL:|replace| SELECT-END IL:|of| SELECTION IL:|with| NIL)
    ;; we use to compute the selection position, but (a) this causes problems because some of these values might not be computed yet, and (b)
    ;; ComputeSelectionPosition should be called anyway.  Here's the old code:
    ;; (replace SelectStartX of selection with (fetch StartX of node))
    ;; (replace SelectStartLine of selection with (fetch FirstLine of node))
    ;; (replace SelectEndX of selection with (IPLUS (fetch StartX of node)
    ;; (fetch ActualLLength of node)))
    ;; (replace SelectEndLine of selection with (fetch LastLine of node))
    (IL:|replace| SELECT-START-X IL:|of| SELECTION IL:|with| NIL)
    (IL:|replace| SELECT-TYPE IL:|of| SELECTION IL:|with| 'STRUCTURE)))
```

(SET-SELECTION-NOWHERE

```
(IL:LAMBDA (SELECTION)                                  ; Edited 17-Nov-87 11:27 by DCB
```

;;; there is no current selection

```
    (IL:REPLACE SELECT-NODE IL:OF SELECTION IL:WITH NIL)))
```

(SHIFT-DOWN

```
  (IL:LAMBDA NIL                                        ; Edited  7-Jul-87 13:04 by DCB
```

;; check which selection modifer keys are held down, and return one of the atoms Move, Copy, Delete, or NIL. The META key is not considered a ;; "selection modifer". It is used to popup the command menu.

```
    (COND
       ((IL:KEYDOWNP 'IL:MOVE)
        'MOVE)
       ((IL:KEYDOWNP 'IL:COPY)
        'COPY)
       ((IL:SHIFTDOWNP 'IL:SHIFT)
        (IF (IL:SHIFTDOWNP 'IL:CTRL)
            'MOVE
            'COPY))
       ((IL:SHIFTDOWNP 'IL:CTRL)
        'DELETE))))
```

(SHOW-CARET

```
  (IL:LAMBDA (CONTEXT COMPUTE-POS? SCROLL?)             ; Edited 13-Jun-88 18:59 by Snow
```

;;; COMMAND is the command name run prior to this update. Normalize the caret if: the user is inside a structure (point-type not STRUCTURE), or ;;; we're specifically told to scroll.

```
    (LET ((CARET-POINT (IL:|fetch| CARET-POINT IL:|of| CONTEXT)))
         (WHEN (IL:|fetch| POINT-NODE IL:|of| CARET-POINT)
             (WHEN COMPUTE-POS? (COMPUTE-POINT-POSITION CARET-POINT CONTEXT))
             (IL:|freplace| CARET IL:|of| CONTEXT IL:|with| (IL:\\CARET.CREATE (IF (EQ (IL:|ffetch| POINT-TYPE IL:|of|
                                                                                                    CARET-POINT
                                                                                                    )
                                                                                'STRUCTURE)
                                                                           STRUCTURE-CARET
                                                                           ATOM-CARET)))
             (WHEN (OR (NOT (EQ (IL:|ffetch| POINT-TYPE IL:|of| CARET-POINT)
                                'STRUCTURE))
                       SCROLL?)
                 ;; AUTO SCROLL: check for caret off screen.
                 (LET* ((WINDOW (IL:|ffetch| DISPLAY-WINDOW IL:|of| CONTEXT))
                        (REGION (IL:DSPCLIPPINGREGION NIL WINDOW))
                        SELECTION CARET-X CARET-Y X-AMOUNT Y-AMOUNT)
                       ;; if its a pending delete point, get the location out of the selection
                       (COND
                          ((IL:TYPE? EDIT-SELECTION (SETQ SELECTION (IL:FFETCH POINT-NODE IL:OF CARET-POINT)))
                           (SETQ CARET-X (IL:FFETCH SELECT-START-X IL:OF SELECTION))
                           (SETQ CARET-Y (IL:FETCH BASE-LINE-Y IL:OF (IL:FFETCH SELECT-START-LINE IL:OF SELECTION)
                                                   )))
                          (T (SETQ CARET-X (IL:FFETCH POINT-X IL:|of| CARET-POINT))
                             (SETQ CARET-Y (IL:FETCH BASE-LINE-Y IL:|of| (IL:FFETCH POINT-LINE IL:|of| CARET-POINT))))
                          )
                       ;; with the fancy formatting of sedit, you can end up off the screen in two dimensions at once, so check horizontally and
                       ;; vertically separately, then do the scroll if need be.
                       (COND
                          ((PLUSP (SETQ X-AMOUNT (- CARET-X (IL:FFETCH (IL:REGION IL:RIGHT) IL:|of| REGION))))
                           ;; fell off right edge
                           (SETQ X-AMOUNT (- (FLOOR (IL:FFETCH (IL:REGION IL:WIDTH) IL:|of| REGION)
                                                    -2)
                                             X-AMOUNT)))
                          ((MINUSP (SETQ X-AMOUNT (- CARET-X (IL:FFETCH (IL:REGION IL:LEFT) IL:|of| REGION))))
                           ;; fell off left edge, scroll right
                           (SETQ X-AMOUNT (- (FLOOR (IL:FFETCH (IL:REGION IL:WIDTH) IL:|of| REGION)
                                                    2)
                                             X-AMOUNT)))
                          (T (SETQ X-AMOUNT 0)))
                       (COND
                          ((MINUSP (SETQ Y-AMOUNT (- CARET-Y (IL:FFETCH (IL:REGION IL:BOTTOM) IL:|of| REGION))))
                           ;; fell off bottom edge
                           (SETQ Y-AMOUNT (- (FLOOR (IL:FFETCH (IL:REGION IL:HEIGHT) IL:|of| REGION)
                                                    2)
                                             Y-AMOUNT)))
                          ((PLUSP (SETQ Y-AMOUNT (- CARET-Y (IL:FFETCH (IL:REGION IL:TOP) IL:|of| REGION))))
                           ;; fell off top edge
                           (SETQ Y-AMOUNT (- (FLOOR (IL:FFETCH (IL:REGION IL:HEIGHT) IL:|of| REGION)
                                                    -2)
                                             Y-AMOUNT)))
                          (T (SETQ Y-AMOUNT 0)))
                       (WHEN (OR (NOT (ZEROP X-AMOUNT))
```

```
                                (NOT (ZEROP Y-AMOUNT)))
                            (IL:SCROLLW WINDOW X-AMOUNT Y-AMOUNT))))))))
```

(SHRINKFN

```
(IL:LAMBDA (WINDOW)                                          ; Edited 5-Dec-90 17:29 by woz
```

;; called by the window system when an SEdit window is shrunk. if it doesn't already have one, give it a pretty icon with an appropriate title. also ;; make sure the command process notices that it should die. grab the context lock here, because it wasn't grabbed by the buttoneventfn.

```
(LET* ((CONTEXT (IL:WINDOWPROP WINDOW 'EDIT-CONTEXT))
       (LOCK (IL:|fetch| CONTEXT-LOCK IL:|of| CONTEXT)))
      (COND
         ((IL:EQMEMB :CLOSE-ON-COMPLETION (IL:|fetch| EDIT-OPTIONS IL:|of| CONTEXT))
          ;; can't shrink, because must be a one-time edit
          (IL:|printout| (GET-PROMPT-WINDOW CONTEXT)
                 T "Can't shrink this SEdit.  Must close when done editing.")
          'IL:DON\'T)
         ((IL:OBTAIN.MONITORLOCK LOCK T)
          (IL:RELEASE.MONITORLOCK LOCK)                      ; release before waking sedit
          (COND
             ((EQ (IL:PROCESSPROP (IL:THIS.PROCESS)
                          'IL:NAME)
                  'IL:MOUSE)
              ;; under the mouse, restart the completion under SEdit
              (AWAKE-COMMAND-PROCESS CONTEXT '(COMPLETE NIL :SHRINK))
              'IL:DON\'T)
             (T (SAVE-WINDOW-REGION CONTEXT :SHRINK (IL:|fetch| ICON-TITLE IL:|of| CONTEXT)
                       (IL:|fetch| EDIT-TYPE IL:|of| CONTEXT)
                       (IL:WINDOWREGION WINDOW))
                (WHEN (NOT (IL:WINDOWPROP WINDOW 'IL:ICON))
                    (IL:WINDOWPROP WINDOW 'IL:ICON (LET ((SHRUNKW (IL:TITLEDICONW TITLED-ICON
                                                                   (IL:|fetch| ICON-TITLE IL:|of| CONTEXT)
                                                                   NIL T)))
                                                     (IL:WINDOWPROP SHRUNKW 'IL:COPYFN 'ICON-COPYFN)
                                                     SHRUNKW))))))
          (T (IL:|printout| (GET-PROMPT-WINDOW CONTEXT)
                    T "Can't shrink.  SEdit is busy.")
             'IL:DON\'T)))))
```

(STRING-OFFSET

```
(IL:LAMBDA (STRING START END FONT STRING? POINT-OR-SELECTION STARTX)
                                                             ; Edited 7-Jul-87 13:04 by DCB
```

;;; compute the x coordinate of a point or selection in a litatom or string. for a point, start is NIL and end is the number of characters before the point. for ;;; a selection, start is the number of characters before the start of the selection, and end is the number of characters before the last character of the ;;; selection. string? specifies that we have to account for string quotes.

```
(IL:FOR J IL:FROM 1 IL:TO END IL:BIND (OFFSET IL:_ 0)
                                      (ESC IL:_ (ESCAPE-CHAR))
                                      K
   IL:FIRST (WHEN STRING?
                 (IL:SETQ OFFSET (IL:CHARWIDTH (IL:CHARCODE IL:\")
                                               FONT)))
   IL:DO (WHEN (EQ J START)
               (IL:REPLACE SELECT-START-X IL:OF POINT-OR-SELECTION IL:WITH (IL:IPLUS OFFSET STARTX)))
         (IL:SETQ K (IL:NTHCHARCODE STRING J))
         (IL:SETQ OFFSET (IL:IPLUS (COND
                                      ((AND STRING? (OR (EQ K (IL:CHARCODE IL:\"))
                                                        (EQ K ESC)))
                                       (IL:IPLUS (IL:CHARWIDTH ESC FONT)
                                              (IL:CHARWIDTH K FONT)))
                                      ((AND STRING? (IL:ILESSP K (IL:CHARCODE IL:SPACE)))
                                       (IL:IPLUS (IL:CHARWIDTH (IL:CHARCODE IL:^ FONT)
                                                               FONT)
                                              (IL:CHARWIDTH (IL:IPLUS K 64)
                                                            FONT)))
                                      (T (IL:CHARWIDTH K FONT)))
                                   OFFSET))
   IL:FINALLY (IF START
                  (IL:REPLACE SELECT-END-X IL:OF POINT-OR-SELECTION IL:WITH (IL:IPLUS OFFSET STARTX))
                  (IL:REPLACE POINT-X IL:OF POINT-OR-SELECTION IL:WITH (IL:IPLUS OFFSET STARTX))))))
```

(TRACK-EXTEND

```
(IL:LAMBDA (CONTEXT WINDOW)                                  ; Edited 24-Nov-87 09:53 by DCB
```

;;; we're extending a selection with the right mouse button. display the resulting selection until the user accepts it by releasing the button. we use ;;; smash.using to copy the contents of one selection into another

```
(IL:FIRST (IL:SETQ PENDING-TYPE NIL)
          ;; extending a selection cancels the point
          (SET-POINT-NOWHERE PENDING-CARET)
          (COND
```

```
        ((IL:FETCH SELECT-NODE IL:OF PENDING-SELECTION)
         (SMASH-USING EDIT-SELECTION INITIAL-SELECTION PENDING-SELECTION))
        ((IL:FETCH SELECT-NODE IL:OF (IL:FETCH SELECTION IL:OF CONTEXT))
         (SMASH-USING EDIT-SELECTION INITIAL-SELECTION (IL:FETCH SELECTION IL:OF CONTEXT)))
        (T ;; there's no selection to extend, so nothing happens.  wait until the mouse button comes up.  this could be changed;  i think it
           ;; would be more convenient if you could extend from a point as well as a selection
           (IL:UNTILMOUSESTATE (NOT IL:RIGHT))
           (RETURN)))
  IL:DO (SMASH-USING EDIT-SELECTION SCRATCH-SELECTION INITIAL-SELECTION)
                                                          ; compute the extended selection
        (EXTEND-SELECTION SCRATCH-SELECTION CONTEXT (IL:LASTMOUSEX WINDOW)
               (IL:LASTMOUSEY WINDOW))
        (WHEN (OR (IL:NEQ (IL:FETCH SELECT-NODE IL:OF PENDING-SELECTION)
                          (IL:FETCH SELECT-NODE IL:OF SCRATCH-SELECTION))
                  (IL:NEQ (IL:FETCH SELECT-START IL:OF PENDING-SELECTION)
                          (IL:FETCH SELECT-START IL:OF SCRATCH-SELECTION))
                  (IL:NEQ (IL:FETCH SELECT-END IL:OF PENDING-SELECTION)
                          (IL:FETCH SELECT-END IL:OF SCRATCH-SELECTION)))
                                                          ; if it's different from the last extended selection, fix the display
            (DISPLAY-SELECTION PENDING-SELECTION WINDOW PENDING-SHIFT)
            (WHEN (NULL (IL:FETCH SELECT-START-X IL:OF SCRATCH-SELECTION))
                  (COMPUTE-SELECTION-POSITION SCRATCH-SELECTION CONTEXT))
            (DISPLAY-SELECTION SCRATCH-SELECTION WINDOW PENDING-SHIFT)
            (SMASH-USING EDIT-SELECTION PENDING-SELECTION SCRATCH-SELECTION))
                                                          ; keep watching for changes in modifier keys
        (CHECK-SELECTION-SHIFT CONTEXT)
        (IL:BLOCK)
  IL:REPEATUNTIL (IL:MOUSESTATE (NOT IL:RIGHT)))))
```

(TRACK-SELECT

```
(IL:LAMBDA (CONTEXT WINDOW)                               ; Edited 24-Nov-87 09:54 by DCB
```

;; we're making a selection with the left or middle mouse button. display the resulting selection until the user accepts it by releasing the button

```
(IL:BIND (POINT-TYPE IL:_ (COND
                          ((IL:LASTMOUSESTATE IL:LEFT)
                           ;; left button select within an atom
                           'ATOM)
                          (T ;; middle button selects structures
                             'STRUCTURE)))
         POINT? BAR-X BAR-Y BAR-LINE BAR-HEIGHT IL:FIRST (WHEN (GROW-CLICK? CONTEXT POINT-TYPE WINDOW)
                                                               ;; if this can be parsed as part of a multi-click sequence to grow the
                                                               ;; current selection, do it
                                                               (WHEN (AND (NOT PENDING-SHIFT)
                                                                          (IL:FETCH SELECT-NODE IL:OF
                                                                                    PENDING-SELECTION
                                                                                    ))
                                                                   (SET-POINT PENDING-CARET CONTEXT
                                                                          (IL:FETCH SELECT-NODE IL:OF
                                                                                    PENDING-SELECTION
                                                                                    )
                                                                          NIL T)
                                                                   (WHEN (IL:FETCH POINT-NODE IL:OF PENDING-CARET)
                                                                         (COMPUTE-POINT-POSITION PENDING-CARET
                                                                                CONTEXT)))
                                                               (RETURN))
                                                         (SMASH-USING EDIT-SELECTION SCRATCH-SELECTION
                                                                PENDING-SELECTION)
```

```
  IL:DO ;; decide where the new point and selection will be
        (PLACE-CARET-AND-SELECTION (AND (NULL PENDING-SHIFT)
                                        PENDING-CARET)
              PENDING-SELECTION CONTEXT (IL:LASTMOUSEX WINDOW)
              (IL:LASTMOUSEY WINDOW)
              POINT-TYPE)
        (WHEN PENDING-SHIFT
            ;; if modifier keys are down we won't set the caret point
            (SET-POINT-NOWHERE PENDING-CARET))
        ;; show a vertical bar where the caret will be placed
        (TRACK-BAR-IN-TRACK-SELECT)
        (WHEN (OR (IL:NEQ (IL:FETCH SELECT-NODE IL:OF PENDING-SELECTION)
                          (IL:FETCH SELECT-NODE IL:OF SCRATCH-SELECTION))
                  (IL:NEQ (IL:FETCH SELECT-START IL:OF PENDING-SELECTION)
                          (IL:FETCH SELECT-START IL:OF SCRATCH-SELECTION))
                  (IL:NEQ (IL:FETCH SELECT-END IL:OF PENDING-SELECTION)
                          (IL:FETCH SELECT-END IL:OF SCRATCH-SELECTION)))
            ;; if this is a new selection, display it
            (DISPLAY-SELECTION SCRATCH-SELECTION WINDOW PENDING-SHIFT)
            (DISPLAY-SELECTION PENDING-SELECTION WINDOW PENDING-SHIFT)
            (SMASH-USING EDIT-SELECTION SCRATCH-SELECTION PENDING-SELECTION))
```

```
            (CHECK-SELECTION-SHIFT CONTEXT)
            (IL:BLOCK)
        IL:REPEATUNTIL (IL:MOUSESTATE (OR IL:UP IL:RIGHT))
        IL:FINALLY (WHEN POINT?
                        ;; take down the vertical bar at the caret position
                        (IL:BLTSHADE IL:BLACKSHADE WINDOW BAR-X BAR-Y 1 BAR-HEIGHT 'IL:INVERT))
                    ;; remember where the mouse is, so that we can detect multi-click sequences
                    (IL:SETQ PENDING-LAST-X (IL:LASTMOUSEX WINDOW))
                    (IL:SETQ PENDING-LAST-Y (IL:LASTMOUSEY WINDOW))
                    (IL:SETQ PENDING-TYPE POINT-TYPE))))
```

(UNDERLINE-SELECTION

```
  (IL:LAMBDA (SELECTION WINDOW SHADE)                    ; Edited 17-Nov-87 11:27 by DCB
```

;;; use draw.underline to underline the this selection with the specified shade

```
    (DRAW-UNDERLINE (IL:FETCH SELECT-START-X IL:OF SELECTION)
           (IL:FETCH SELECT-START-LINE IL:OF SELECTION)
           (IL:FETCH SELECT-END-X IL:OF SELECTION)
           (IL:FETCH SELECT-END-LINE IL:OF SELECTION)
          WINDOW SHADE)))
```

(UPDATE-TITLE

```
  (IL:LAMBDA (CONTEXT WINDOW ALWAYS?)                    ; Edited  7-Jul-87 13:04 by DCB
```

;;; MUST BE CALLED UNDER SEDIT'S PROFILE: Expects *PACKAGE* to be bound properly. Update the window title to reflect the state of the edit.
;;; toggle the asterisk that means "unsaved changes", fixup the current package...

;;; The OR to test if any field has changed is okay because only one thing can happen at a time, and so only one of the or clauses can be true on any
;;; call to this function.

```
    (LET ((TITLE-INFO (IL:WINDOWPROP WINDOW 'TITLE-INFO))
          (CHANGED-STRUCTURE (IL:FETCH CHANGED-STRUCTURE? IL:OF CONTEXT))
          (NAME (IL:FETCH ICON-TITLE IL:OF CONTEXT)))
         (WHEN (OR (WHEN (IL:NEQ CHANGED-STRUCTURE (IL:LISTGET TITLE-INFO :|ChangedStructure?|))
                       (IL:LISTPUT TITLE-INFO :|ChangedStructure?| CHANGED-STRUCTURE)
                       T)
                   (WHEN (IL:NEQ *PACKAGE* (IL:LISTGET TITLE-INFO :|package|))
                       (IL:LISTPUT TITLE-INFO :|package| *PACKAGE*)
                       T)
                   (WHEN (IL:NEQ NAME (IL:LISTGET TITLE-INFO :|name|))
                       (IL:LISTPUT TITLE-INFO :|name| NAME)
                       T)
                  ALWAYS?)
             (IL:WINDOWPROP WINDOW 'IL:TITLE (IL:CONCAT (IF CHANGED-STRUCTURE
                                                            "* "
                                                            "")
                                                        EDITOR-NAME " " (OR NAME "")
                                                        "  Package: "
                                                        (PACKAGE-NAME *PACKAGE*)))))))

)

(IL:PUTPROPS IL:SEDIT-WINDOW IL:COPYRIGHT ("Venue & Xerox Corporation" 1986 1987 1988 1990 1991 1992 2018))
```

FUNCTION INDEX

VARIABLE INDEX

MACRO INDEX

PROPERTY INDEX