File created: 17-May-90 11:06:11 **{DSK}<usr>local>lde>lispcore>sources>SEDIT-LINEAR.;2**

changes to: (IL:VARS IL:SEDIT-LINEARCOMS)

previous date: 13-Apr-88 11:51:14 {DSK}<usr>local>lde>lispcore>sources>SEDIT-LINEAR.;1

Read Table: XCL

Package: SEDIT

Format: XCCS


```
(IL:RPAQQ IL:SEDIT-LINEARCOMS
          ((IL:PROP IL:FILETYPE IL:SEDIT-LINEAR)
           (IL:PROP IL:MAKEFILE-ENVIRONMENT IL:SEDIT-LINEAR)
           (IL:LOCALVARS . T)
           (IL:DECLARE\: IL:DONTCOPY IL:DOEVAL@COMPILE (IL:FILES IL:SEDIT-DECLS))
           (IL:FUNCTIONS CLEAR-ALL-LINEAR-FORMS CLEAR-LINEAR-FORM RELINEARIZE-PRELINEARIZED-NODE)
           (IL:FNS CLEAN-UP-AFTER-RELINEARIZATION FIRST-LINE-LINEAR GENERATE-LINEAR-FORM LAST-LINE-LINEAR
                  LINE-FINISHED LINEAR-ITEM-WIDTH LINEARIZE NEW-BLOCK NEXT-LINEAR-ITEM OUTPUT-BITMAP
                  OUTPUT-CONSTANT-STRING OUTPUT-CR OUTPUT-SPACE OUTPUT-STRING PAINT-TO-END-OF-LINE
                  RECOMPUTE-FORMAT-VALUES RELINEARIZE REPAINT REUSE-LINEAR-FORM SHIFT-BLOCK TRY-REUSING-BITS)))

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

(IL:PUTPROPS IL:SEDIT-LINEAR 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)
)


(DEFUN CLEAR-ALL-LINEAR-FORMS (CONTEXT)
   (WALK-UP-TREE (SUBNODE 1 (IL:|fetch| ROOT IL:|of| CONTEXT))
          CONTEXT
          #'CLEAR-LINEAR-FORM))


(DEFUN CLEAR-LINEAR-FORM (NODE)

;;; throw away old linear form (and create new one if a prelinearized node)

   (COND
      ((IL:|fetch| LINEARIZE IL:|of| (IL:|fetch| NODE-TYPE IL:|of| NODE))
       (IL:|replace| START-X IL:|of| NODE IL:|with| 0)
       (IL:|replace| LINEAR-FORM IL:|of| NODE IL:|with| (CREATE-WEAK-LINK NODE)))
      (T (RELINEARIZE-PRELINEARIZED-NODE NODE)))
   (IL:|replace| LINEAR-THREAD IL:|of| NODE IL:|with| NIL))


(DEFUN RELINEARIZE-PRELINEARIZED-NODE (NODE)

;;; we've changed a prelinearized node.  fix up the width estimates

   (LET ((LITEM (CAR (IL:|fetch| LINEAR-FORM IL:|of| NODE))))
        (WHEN (TYPEP LITEM 'STRING-ITEM)
            (LET ((NEW-WIDTH (STRINGWIDTH (IL:|fetch| STRING IL:|of| LITEM)
                                          (IL:|fetch| FONT IL:|of| LITEM)
                                          (IL:|fetch| PRIN-2? IL:|of| LITEM))))
                 (IL:|replace| WIDTH IL:|of| LITEM IL:|with| NEW-WIDTH)
                 (IL:|replace| INLINE-WIDTH IL:|of| NODE IL:|with| NEW-WIDTH)
                 (IL:|replace| PREFERRED-WIDTH IL:|of| NODE IL:|with| NEW-WIDTH)
                 (IL:|replace| ACTUAL-WIDTH IL:|of| NODE IL:|with| NEW-WIDTH)
                 (IL:|replace| ACTUAL-LLENGTH IL:|of| NODE IL:|with| NEW-WIDTH)))))

(IL:DEFINEQ

(CLEAN-UP-AFTER-RELINEARIZATION
  (IL:LAMBDA (CONTEXT NODE FOLLOWING-LINE Y-1 Y-2)                    ; Edited 17-Nov-87 11:37 by DCB

;;; we've just finished relinearizing this node.  adjust the y coordinates of everything that follows, and fix up the rest of the window.  y1 and y2 record
;;; where the following lines start and how far they must be shifted

    (LET* ((WINDOW (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT))
           (EXTENT (IL:WINDOWPROP WINDOW 'IL:EXTENT))
           (BOTTOM (IL:FETCH WINDOW-BOTTOM IL:OF CONTEXT))
           (LEFT (IL:FETCH WINDOW-LEFT IL:OF CONTEXT))
           (WINDOW-WIDTH (IL:ADD1 (IL:IDIFFERENCE (IL:FETCH WINDOW-RIGHT IL:OF CONTEXT)
                                                  LEFT)))
```

```
          DELTA RY-1 RY-2)
         (COND
            (FOLLOWING-LINE
                    ;; there are lines after this node.  fix up the links, adjust their y coordinates, and fix the window's extent
                    ;; used to be replace PrevLine of following.line with (fetch LastLineLinear of node)
                    (IL:REPLACE PREV-LINE IL:OF FOLLOWING-LINE IL:WITH (LAST-LINE-LINEAR NODE CONTEXT))
                    (IL:SETQ DELTA (IL:IDIFFERENCE (IL:FETCH NEXT-LINE-Y IL:OF (IL:FETCH LAST-LINE IL:OF NODE))
                                   (IL:FETCH YCOORD IL:OF FOLLOWING-LINE)))
                    (WHEN (NOT (EQ 0 DELTA))
                        (IL:BIND (LINE IL:_ FOLLOWING-LINE) IL:DO (IL:ADD (IL:FETCH YCOORD IL:OF LINE)
                                                                          DELTA)
                           IL:REPEATWHILE (IL:SETQ LINE (CAR (IL:FETCH NEXT-LINE IL:OF LINE))))
                        (IL:ADD (IL:FETCH (IL:REGION IL:BOTTOM) IL:OF EXTENT)
                               DELTA)
                        (IL:ADD (IL:FETCH (IL:REGION IL:HEIGHT) IL:OF EXTENT)
                               (IL:IMINUS DELTA)))
                    ;; if the following stuff isn't off the bottom of the window, we'll have to fix it up
                    (WHEN (AND Y-2 (IL:IGEQ Y-2 BOTTOM))
                        (COND
                           ((IL:IGEQ Y-2 (IL:FETCH WINDOW-TOP IL:OF CONTEXT))
                            ;; none of the changes were visible, so we just have to twiddle the coordinate system
                            (IL:WYOFFSET (IL:IDIFFERENCE Y-1 Y-2)
                                   WINDOW))
                           ((IL:NEQ Y-1 Y-2)
                            (COND
                               ((IL:IGREATERP Y-1 Y-2)
                                ;; bitblt the following lines up, and repaint below them
                                (IL:SETQ RY-1 Y-2)
                                (IL:SETQ RY-2 (IL:IPLUS Y-2 1 (IL:IDIFFERENCE Y-2 Y-1)))
                                (IF (IL:IGREATERP RY-2 BOTTOM)
                                    (IL:BITBLT WINDOW LEFT (IL:IPLUS (IL:IDIFFERENCE Y-1 Y-2)
                                                                     BOTTOM)
                                           WINDOW LEFT BOTTOM WINDOW-WIDTH (IL:IDIFFERENCE RY-2 BOTTOM))
                                    (IL:SETQ RY-2 BOTTOM)))
                               (T ;; the following stuff moves down, if it hasn't already been overwritten
                                  (COND
                                     ((IL:IGEQ Y-1 BOTTOM)
                                      (IL:SETQ RY-1 (IL:FETCH YCOORD IL:OF FOLLOWING-LINE))
                                      (IL:SETQ RY-2 (IL:IPLUS (IL:SUB1 BOTTOM)
                                                              (IL:IDIFFERENCE Y-2 Y-1)))
                                      (IL:BIND (NEXT-LINE IL:_ FOLLOWING-LINE)
                                             NEXT-LINE-Y
                                         IL:WHILE (AND NEXT-LINE (IL:IGEQ (IL:SETQ NEXT-LINE-Y
                                                                                   (IF (IL:SETQ NEXT-LINE
                                                                                         (CAR (IL:FETCH NEXT-LINE
                                                                                                        IL:OF NEXT-LINE)))
                                                                                       (IL:FETCH YCOORD IL:OF NEXT-LINE)
                                                                                       (IL:FETCH NEXT-LINE-Y IL:OF
                                                                                                                  FOLLOWING-LINE
                                                                                             )))
                                                                           RY-2))
                                         IL:DO (IL:SETQ FOLLOWING-LINE NEXT-LINE)
                                               (IL:SETQ RY-1 NEXT-LINE-Y))
                                      (IL:SETQ RY-2 (IL:IPLUS RY-1 (IL:IDIFFERENCE Y-1 Y-2)))
                                      (IL:BITBLT WINDOW LEFT (IL:ADD1 RY-2)
                                             WINDOW LEFT (IL:ADD1 RY-1)
                                             WINDOW-WIDTH
                                             (IL:IDIFFERENCE Y-1 RY-2)))
                                     (T (IL:SETQ RY-1 Y-2)))
                                  (IL:SETQ RY-2 BOTTOM)))))
                            ;; now that we've figured out what needs to be repainted, blank the area and repaint it
                            (IL:BLTSHADE IL:WHITESHADE WINDOW LEFT RY-2 WINDOW-WIDTH (IL:ADD1 (IL:IDIFFERENCE
                                                                                                RY-1 RY-2)))
                            (WHEN FOLLOWING-LINE
                                (REPAINT CONTEXT (IL:FETCH INDENT IL:OF FOLLOWING-LINE)
                                       (IL:FETCH BASE-LINE-Y IL:OF FOLLOWING-LINE)
                                       (CDR (IL:FETCH NEXT-LINE IL:OF (CAR (IL:FETCH PREV-LINE IL:OF
                                                                                                FOLLOWING-LINE
                                                                                   ))))
                                       RY-2))))))
            (T ;; there's nothing after the relinearized material -- blank anything after it on the window and adjust the extent
               (WHEN (AND Y-2 (IL:IGREATERP Y-2 Y-1))
                   (IL:BLTSHADE IL:WHITESHADE WINDOW LEFT Y-1 WINDOW-WIDTH (IL:IDIFFERENCE Y-2 Y-1)))
               (IL:SETQ DELTA (IL:FETCH NEXT-LINE-Y IL:OF (IL:FETCH LAST-LINE IL:OF NODE)))
               (IL:REPLACE (IL:REGION IL:BOTTOM) IL:OF EXTENT IL:WITH (IL:ADD1 DELTA))
               (IL:REPLACE (IL:REGION IL:HEIGHT) IL:OF EXTENT IL:WITH (IL:IMINUS DELTA)))))))
```

(**FIRST-LINE-LINEAR**
  (IL:LAMBDA (NODE CONTEXT) ; Edited 17-Nov-87 11:38 by DCB

```
;;; find the info which used to be stored in FirstLineLinear (i.e. the tail of the linear form beginning with the first line of this node). we try to step back one
;;; more line and then forward; if this is the first line we know it must be the beginning of the root's linear form

    (AND (IL:SETQ NODE (IL:FETCH FIRST-LINE IL:OF NODE))
         (IF (IL:FETCH PREV-LINE IL:OF NODE)
             (IL:FETCH NEXT-LINE IL:OF (CAR (IL:FETCH PREV-LINE IL:OF NODE)))
             (IL:FETCH LINEAR-FORM IL:OF (IL:FETCH ROOT IL:OF CONTEXT))))))
```

(GENERATE-LINEAR-FORM

```
  (IL:LAMBDA (NODE CONTEXT RIGHT-MARGIN)                    ; Edited 7-Apr-88 11:02 by woz

;;; we need to compute the linear form of this node. if there's a previously computed linear form and it fits the constraints, we can just reuse it; otherwise
;;; call the Linearize method for the node

    (LET ((CURRENT-X (IL:|fetch| CURRENT-X IL:|of| CONTEXT))
          (LINEARIZE (IL:|fetch| LINEARIZE IL:|of| (IL:|fetch| NODE-TYPE IL:|of| NODE))))
        ;; this next IF test is ugly. want to reuse the LF if we can. Don't try to reuse the root's LF, becuase it starts with a line start which messes up
        ;; scan.for.bounds, and the root has no bits on the screen so there's no savings in reusing it. Otherwise check if it has changed, and if it will
        ;; fit is the new space provided (we might be reshaping), and if all those pass, then reuse the LF.
        (COND
           ((OR (NULL LINEARIZE)
                (AND (IL:NEQ NODE (IL:|fetch| ROOT IL:|of| CONTEXT))
                     (IL:NEQ (IL:|fetch| START-X IL:|of| NODE)
                             0)
                     (NOT (IL:|fetch| CHANGED? IL:|of| NODE))
                     (OR (IL:ILEQ (IL:IPLUS CURRENT-X (IL:|fetch| ACTUAL-WIDTH IL:|of| NODE))
                                  RIGHT-MARGIN)
                         (IL:ILEQ (IL:IDIFFERENCE (IL:|fetch| RIGHT-MARGIN IL:|of| NODE)
                                                  (IL:|fetch| START-X IL:|of| NODE))
                                  (IL:IDIFFERENCE RIGHT-MARGIN CURRENT-X)))
                     (OR (IL:|fetch| INLINE? IL:|of| NODE)
                         (IL:IGEQ (IL:IDIFFERENCE (IL:|fetch| RIGHT-MARGIN IL:|of| NODE)
                                                  (IL:|fetch| START-X IL:|of| NODE))
                                  (IL:IDIFFERENCE RIGHT-MARGIN CURRENT-X)))))
            ;; the old linear form will do
            (IL:|replace| RIGHT-MARGIN IL:|of| NODE IL:|with| RIGHT-MARGIN)
            (REUSE-LINEAR-FORM NODE CONTEXT))
           (T                                                   ; we've got to call the Linearize method. initialize various
                                                                ; random fields and do it
              (IL:|replace| START-X IL:|of| NODE IL:|with| CURRENT-X)
              (IL:|replace| RIGHT-MARGIN IL:|of| NODE IL:|with| RIGHT-MARGIN)
              (IL:|replace| ACTUAL-WIDTH IL:|of| NODE IL:|with| 0)
              (IL:|replace| FIRST-LINE IL:|of| NODE IL:|with| (CAR (IL:|fetch| CURRENT-LINE IL:|of| CONTEXT)))
              (IL:|replace| CURRENT-NODE IL:|of| CONTEXT IL:|with| NODE)
              (IL:|replace| LAST-LINEARIZED-SUB-NODE-INDEX IL:|of| CONTEXT IL:|with| 0)
              (COND
                 ((IL:|fetch| SUPER-NODE IL:|of| NODE)          ; setup pointers to start at the beginning of this nodes linear form.
                  (IL:|replace| LINEAR-POINTER IL:|of| CONTEXT IL:|with| (IL:|fetch| LINEAR-FORM IL:|of| NODE))
                  (IL:|replace| LINEAR-PREV IL:|of| CONTEXT IL:|with| NODE))
                 (T ;; (hack) the linear form of the root doesn't correspond to what linearize.root will produce, since it has the initial line start as
                    ;; its first element. this should be fixed, but in the meantime we'll just skip over it

                    ;; SO: here the linear form of the root is alread set to a list of the first line start and a weak-link. make the linear-pointer point
                    ;; into the list at the weak-link.
                    (IL:|replace| LINEAR-POINTER IL:|of| CONTEXT IL:|with| (CDR (IL:|replace| LINEAR-PREV IL:|of| CONTEXT
                                                                                  IL:|with| (IL:|fetch| LINEAR-FORM
                                                                                             IL:|of| NODE))))))
              (IL:|replace| ACTUAL-LLENGTH IL:|of| NODE IL:|with| NIL)
              (FUNCALL LINEARIZE NODE CONTEXT)
              ;; now we're done with this node (and thus its subnodes), so move back to its super.
              (IL:|replace| CURRENT-NODE IL:|of| CONTEXT IL:|with| (IL:|fetch| SUPER-NODE IL:|of| NODE))
              (WHEN (NOT (AND (IL:|type?| WEAK-LINK (IL:|fetch| LINEAR-POINTER IL:|of| CONTEXT))
                              (EQ (IL:FETCH DESTINATION IL:OF (IL:|fetch| LINEAR-POINTER IL:|of| CONTEXT))
                                  NODE)))
                  ;; we should have finished linearizing the node, and so linear-pointer will be at the weak-link. if not, (i guess it didn't need to be
                  ;; relinearized? and thus the matching? test below?) set it there so we can go on.
                  (SET-LINEAR CONTEXT (CDR (LAST (IL:FETCH LINEAR-FORM IL:OF NODE))))
                  (WHEN (IL:|fetch| MATCHING? IL:|of| CONTEXT)
                        (NEW-BLOCK CONTEXT)
                        (IL:|replace| MATCHING? IL:|of| CONTEXT IL:|with| NIL))
                                                                ; used to be replace LastLineLinear of x with (fetch CurrentLine
                                                                ; of context)
              (IL:|replace| LAST-LINE IL:|of| NODE IL:|with| (CAR (IL:|fetch| CURRENT-LINE IL:|of| CONTEXT)))
              (IL:|replace| ACTUAL-WIDTH IL:|of| NODE IL:|with| (IL:IDIFFERENCE (IL:IMAX (IL:|fetch| ACTUAL-WIDTH
                                                                                          IL:|of| NODE)
                                                                                        (IL:|fetch| CURRENT-X IL:|of| CONTEXT)
                                                                                        )
                                                                                 CURRENT-X))
              (IL:|replace| ACTUAL-LLENGTH IL:|of| NODE IL:|with| (IL:IDIFFERENCE (IL:|fetch| CURRENT-X IL:|of| CONTEXT)
                                                                                  CURRENT-X)))))
    (IL:|replace| CHANGED? IL:|of| NODE IL:|with| NIL)))
```

(LAST-LINE-LINEAR

```
(IL:LAMBDA (NODE CONTEXT)                                    ; Edited  7-Jul-87 12:47 by DCB
```

;; find the info which used to be stored in LastLineLinear (i.e. the tail of the linear form beginning with the last line of this node). we try to step back
;; one more line and then forward; if there is no previous line then the whole linear form must be on one line, and thus the last line is simply the
;; root's linear form

```
(AND (IL:SETQ NODE (IL:FETCH LAST-LINE IL:OF NODE))
     (IF (IL:FETCH PREV-LINE IL:OF NODE)
         (IL:FETCH NEXT-LINE IL:OF (CAR (IL:FETCH PREV-LINE IL:OF NODE)))
         (IL:FETCH LINEAR-FORM IL:OF (IL:FETCH ROOT IL:OF CONTEXT))))))
```

(LINE-FINISHED

```
(IL:LAMBDA (CONTEXT X LINEAR FORCE)                          ; Edited 17-Nov-87 11:38 by DCB
```

;;; we've finished a line which is visible (or above the window) we only flush it if it reuses bits or we're forced

```
(WHEN (IL:FETCH BELOW? IL:OF CONTEXT)
      (IL:SHOULDNT "tried to flush a line off the bottom of the screen"))
(LET ((THIS-LINE (CAR (IL:FETCH CURRENT-LINE IL:OF CONTEXT))))
     (WHEN (IL:ILESSP (IL:FETCH NEXT-LINE-Y IL:OF THIS-LINE)
                      (IL:FETCH WINDOW-BOTTOM IL:OF CONTEXT))
```

;; this is the last line visible in the window. force it and don't come back

```
         (IL:REPLACE BELOW? IL:OF CONTEXT IL:WITH T)
         (IL:SETQ FORCE T))
     (COND
        ((IL:ILESSP (IL:FETCH NEXT-LINE-Y IL:OF THIS-LINE)
                    (IL:FETCH WINDOW-TOP IL:OF CONTEXT))
```

;; it's visible. fix up the block list, and then check if any of them can reuse bits visible in the window

```
         (IL:REPLACE BLOCK-WIDTH IL:OF (IL:FETCH CURRENT-BLOCK IL:OF CONTEXT)
            IL:WITH (IL:IDIFFERENCE X (IL:FETCH BLOCK-NEW-X IL:OF (IL:FETCH CURRENT-BLOCK IL:OF CONTEXT))))
         (IL:REPLACE BLOCK-START IL:OF (OR (IL:FETCH NEXT-BLOCK IL:OF (IL:FETCH CURRENT-BLOCK IL:OF CONTEXT))
                                           (IL:REPLACE NEXT-BLOCK IL:OF (IL:FETCH CURRENT-BLOCK IL:OF CONTEXT)
                                              IL:WITH (IL:CREATE LINE-BLOCK)))
            IL:WITH LINEAR)
         (WHEN (IL:TYPE? LINE-START (CAR (IL:FETCH BLOCK-START IL:OF (IL:FETCH FIRST-BLOCK IL:OF CONTEXT))))
             (IL:REPLACE BLOCK-START IL:OF (IL:FETCH FIRST-BLOCK IL:OF CONTEXT)
                IL:WITH (CDR (IL:FETCH BLOCK-START IL:OF (IL:FETCH FIRST-BLOCK IL:OF CONTEXT)))))
         (IL:FOR (BLOCK IL:_ (IL:FETCH FIRST-BLOCK IL:OF CONTEXT)) IL:BY (IL:FETCH NEXT-BLOCK IL:OF BLOCK)
            IL:DO (WHEN (AND (IL:FETCH BITS? IL:OF BLOCK)
                             (IL:TRY-REUSING-BITS CONTEXT BLOCK))
```

;; found some bits we can reuse, so paint up to this point and make sure we dump the rest of this line

```
                      (IL:SETQ FORCE T))
            IL:REPEATUNTIL (EQ BLOCK (IL:FETCH CURRENT-BLOCK IL:OF CONTEXT))
            IL:FINALLY (WHEN FORCE
```

;; display the rest

```
                           (IL:PAINT-TO-END-OF-LINE CONTEXT LINEAR)
                           (WHEN LINEAR
```

;; that wasn't the last line, so set up for the next

```
                                (IL:REPLACE REPAINT-START IL:OF CONTEXT IL:WITH LINEAR)
                                (IL:REPLACE REPAINT-LINE IL:OF CONTEXT IL:WITH (CAR LINEAR))
                                (IL:REPLACE REPAINT-X IL:OF CONTEXT IL:WITH (IL:FETCH INDENT IL:OF (CAR LINEAR))
                                      )
                                (WHEN (AND (IL:FETCH MATCHING? IL:OF CONTEXT)
                                           (NOT (IL:FETCH VISIBLE? IL:OF CONTEXT))
                                           (IL:ILEQ (IL:FETCH OLD-BOTTOM IL:OF (CAR LINEAR))
                                                  (IL:FETCH WINDOW-TOP IL:OF CONTEXT)))
```

;; we were off the top of the screen, but now we're on

```
                                    (IL:REPLACE VISIBLE? IL:OF CONTEXT IL:WITH T))))))
        (LINEAR
```

;; when it's off the top of the window, we just have to reset things (unless it was the last)

```
               (IL:REPLACE REPAINT-START IL:OF CONTEXT IL:WITH LINEAR)
               (IL:REPLACE REPAINT-LINE IL:OF CONTEXT IL:WITH (CAR LINEAR))
               (IL:REPLACE REPAINT-X IL:OF CONTEXT IL:WITH (IL:FETCH INDENT IL:OF (CAR LINEAR)))
               (WHEN (AND (IL:FETCH MATCHING? IL:OF CONTEXT)
                          (NOT (IL:FETCH VISIBLE? IL:OF CONTEXT))
                          (IL:ILEQ (IL:FETCH OLD-BOTTOM IL:OF (CAR LINEAR))
                                   (IL:FETCH WINDOW-TOP IL:OF CONTEXT)))
                     (IL:REPLACE VISIBLE? IL:OF CONTEXT IL:WITH T)))))))
```

(LINEAR-ITEM-WIDTH

```
(IL:LAMBDA (ITEM)                                            ; Edited 17-Nov-87 11:39 by DCB
```

;;; determine the amount of horizontal space taken up by this linear form item

```
(COND
   ((IL:FIXP ITEM)
```

```
        ITEM)
       ((IL:TYPE? STRING-ITEM ITEM)
        (IL:FETCH WIDTH IL:OF ITEM))
       ((IL:LISTP ITEM)
        (IL:BITMAPWIDTH (CDR ITEM)))
       (T (IL:SHOULDNT "this doesn't have a linear width")))))
```

(LINEARIZE

```
(IL:LAMBDA (NODE CONTEXT RIGHT-MARGIN)                    ; Edited 13-Apr-88 10:38 by woz

  ;; fill in the linear form of this node.  make sure that we're actually running as an editor (not just a pretty printer)

  (COND
     ((IL:|fetch| RELINEARIZATION-TIME-STAMP IL:|of| CONTEXT)        ; we're actually editing
      (PROG ((SUPER-NODE (IL:|fetch| CURRENT-NODE IL:|of| CONTEXT)))
            (WHEN (OR (IL:NEQ SUPER-NODE (IL:|fetch| SUPER-NODE IL:|of| NODE))
                      (IL:ILEQ (IL:|fetch| SUB-NODE-INDEX IL:|of| NODE)
                               (IL:|fetch| LAST-LINEARIZED-SUB-NODE-INDEX IL:|of| CONTEXT)))
                  (IL:SHOULDNT "this node shouldn't be linearized now"))
            (WHEN (AND (IL:|fetch| MATCHING? IL:|of| CONTEXT)
                       (NEXT-LINEAR CONTEXT NODE))            ; we're already matching -- all's cool.  fix up the LinearThread in
                                                              ; case it's been smashed.
                  (IL:|replace| LINEAR-THREAD IL:|of| NODE IL:|with| (IL:|fetch| LINEAR-POINTER IL:|of| CONTEXT))
                  (GO OK))
            (COND
               ((IL:|fetch| LINEAR-THREAD IL:|of| NODE)       ; was already linearized -- skip to the appropriate point in the
                                                              ; super's linear form
                (SET-LINEAR CONTEXT (IL:|fetch| LINEAR-THREAD IL:|of| NODE)))
               (T                                             ; insert this node in the super's linear form
                  (SET-LINEAR CONTEXT (CONS (CREATE-WEAK-LINK NODE)
                                            (IL:|fetch| LINEAR-POINTER IL:|of| CONTEXT)))
                  (IL:|replace| LINEAR-THREAD IL:|of| NODE IL:|with| (IL:|fetch| LINEAR-POINTER IL:|of| CONTEXT))))
            (WHEN (IL:|fetch| MATCHING? IL:|of| CONTEXT)      ; we were matching, but lost -- start a new block
                  (NEW-BLOCK CONTEXT)
                  (IL:|replace| MATCHING? IL:|of| CONTEXT IL:|with| NIL))
            (WHEN (AND (NOT (IL:|fetch| BELOW? IL:|of| CONTEXT))
                       (IL:NEQ (IL:|fetch| START-X IL:|of| NODE)
                               0)
                       (IL:IGEQ (IL:|fetch| OLD-TOP IL:|of| (IL:|fetch| FIRST-LINE IL:|of| NODE))
                                (IL:|fetch| WINDOW-BOTTOM IL:|of| CONTEXT))
                       (OR (NOT (IL:|fetch| CHANGED? IL:|of| NODE))
                           (IL:|fetch| LINEARIZE IL:|of| (IL:|fetch| NODE-TYPE IL:|of| NODE))))
                                                              ; we can start matching
                  (IL:|replace| MATCHING? IL:|of| CONTEXT IL:|with| T)
                  (WHEN (OR (IL:|replace| VISIBLE? IL:|of| CONTEXT IL:|with| (AND (IL:ILEQ (IL:|fetch| START-X IL:|of| NODE)
                                                                                           (IL:|fetch| WINDOW-RIGHT
                                                                                              IL:|of| CONTEXT))
                                                                                 (IL:ILEQ (IL:|fetch| OLD-BOTTOM
                                                                                             IL:|of| (IL:|fetch| FIRST-LINE
                                                                                                        IL:|of| NODE))
                                                                                          (IL:|fetch| WINDOW-TOP
                                                                                             IL:|of| CONTEXT))))
                            T)                                ; the stuff we're matching is visible, so build a block describing it
                        (NEW-BLOCK CONTEXT)
                        (LET ((BLOCK (IL:|fetch| CURRENT-BLOCK IL:|of| CONTEXT))
                              (LINE (IL:|fetch| FIRST-LINE IL:|of| NODE)))
                             (IL:|replace| BITS? IL:|of| BLOCK IL:|with| T)
                             (IL:|replace| BLOCK-X IL:|of| BLOCK IL:|with| (IL:|fetch| START-X IL:|of| NODE))
                             (COND
                                ((EQ (IL:|fetch| CACHE-TIME IL:|of| LINE)
                                     (IL:|fetch| RELINEARIZATION-TIME-STAMP IL:|of| CONTEXT))
                                 (IL:|replace| BLOCK-BASE-LINE IL:|of| BLOCK IL:|with| (IL:|fetch| CACHED-Y IL:|of| LINE))
                                 (IL:|replace| BLOCK-ASCENT IL:|of| BLOCK IL:|with| (IL:|fetch| CACHED-ASCENT IL:|of| LINE))
                                 (IL:|replace| BLOCK-DESCENT IL:|of| BLOCK IL:|with| (IL:|fetch| CACHED-DESCENT IL:|of| LINE)))
                                (T (IL:|replace| BLOCK-BASE-LINE IL:|of| BLOCK IL:|with| (IL:|fetch| BASE-LINE-Y IL:|of| LINE))
                                   (IL:|replace| BLOCK-ASCENT IL:|of| BLOCK IL:|with| (IL:|fetch| LINE-ASCENT IL:|of| LINE))
                                   (IL:|replace| BLOCK-DESCENT IL:|of| BLOCK IL:|with| (IL:|fetch| LINE-DESCENT IL:|of| LINE))))
                             )))
         OK                                                   ; we're ready to actually construct/check the linear form
            (GENERATE-LINEAR-FORM NODE CONTEXT (OR RIGHT-MARGIN (IL:|fetch| RIGHT-MARGIN IL:|of| SUPER-NODE)))
            (IL:|replace| LINEAR-PREV IL:|of| CONTEXT IL:|with| (IL:|fetch| LINEAR-THREAD IL:|of| NODE))
            (IL:|replace| LINEAR-POINTER IL:|of| CONTEXT IL:|with| (CDR (IL:|fetch| LINEAR-PREV IL:|of| CONTEXT)))
            (IL:|replace| LAST-LINEARIZED-SUB-NODE-INDEX IL:|of| CONTEXT IL:|with| (IL:|fetch| SUB-NODE-INDEX
                                                                                       IL:|of| NODE))
            (IL:|replace| ACTUAL-WIDTH IL:|of| SUPER-NODE IL:|with| (IL:IMAX (IL:|fetch| ACTUAL-WIDTH IL:|of| SUPER-NODE)
                                                                              (IL:IPLUS (IL:|fetch| START-X IL:|of| NODE)
                                                                                        (IL:|fetch| ACTUAL-WIDTH IL:|of| NODE))))
            (RETURN (IL:|fetch| INLINE? IL:|of| NODE))))
     (T                                                       ; we're pretty printing -- just call the Linearize method, or use the
                                                              ; fixed linear form
        (IL:|replace| RIGHT-MARGIN IL:|of| NODE IL:|with| (OR RIGHT-MARGIN (IL:|fetch| RIGHT-MARGIN
                                                                               IL:|of| (IL:|fetch| SUPER-NODE IL:|of| NODE))))
        (IL:|replace| START-X IL:|of| NODE IL:|with| (IL:|fetch| CURRENT-X IL:|of| CONTEXT))
        (COND
           ((IL:|fetch| LINEARIZE IL:|of| (IL:|fetch| NODE-TYPE IL:|of| NODE))
            (LET ((ME (IL:|fetch| CURRENT-NODE IL:|of| CONTEXT)))
                 (IL:|replace| CURRENT-NODE IL:|of| CONTEXT IL:|with| NODE)
```

```
                (FUNCALL (IL:|fetch| LINEARIZE IL:|of| (IL:|fetch| NODE-TYPE IL:|of| NODE))
                       NODE CONTEXT)
                (IL:|replace| CURRENT-NODE IL:|of| CONTEXT IL:|with| ME)))
          (T                                                    ; this node has a fix linear form (i.e. it's been prelinearized) so
                                                                ; just output it
             (OUTPUT-CONSTANT-STRING CONTEXT (CAR (IL:|fetch| LINEAR-FORM IL:|of| NODE)))))))))
```

(NEW-BLOCK

```
(IL:LAMBDA (CONTEXT)                                            ; Edited 8-Jul-87 17:36 by DCB
```

;;; start a new block in the block list describing this line (we've started or stopped matching)

```
(LET ((BLOCK (IL:FETCH CURRENT-BLOCK IL:OF CONTEXT))
      (X (IL:FETCH CURRENT-X IL:OF CONTEXT)))
     (WHEN (IL:NEQ X (IL:FETCH BLOCK-NEW-X IL:OF BLOCK))
         ;; the current one is non empty, so we need a new one.  fill in the width of the current one before we move on.  if there isn't already a
         ;; next block we'll have to create one
         (IL:REPLACE BLOCK-WIDTH IL:OF BLOCK IL:WITH (IL:IDIFFERENCE X (IL:FETCH BLOCK-NEW-X IL:OF BLOCK)))
         (IL:REPLACE CURRENT-BLOCK IL:OF CONTEXT IL:WITH (IL:SETQ BLOCK (OR (IL:FETCH NEXT-BLOCK IL:OF BLOCK)
                                                                            (IL:REPLACE NEXT-BLOCK
                                                                               IL:OF BLOCK IL:WITH (IL:CREATE
                                                                                                    LINE-BLOCK)
                                                                                                    ))))
         (IL:REPLACE BLOCK-NEW-X IL:OF BLOCK IL:WITH X))
     (IL:REPLACE BLOCK-START IL:OF BLOCK IL:WITH (IL:FETCH LINEAR-POINTER IL:OF CONTEXT))
     (IL:REPLACE BITS? IL:OF BLOCK IL:WITH NIL))))
```

(NEXT-LINEAR-ITEM

```
(IL:LAMBDA (LINEAR)                                             ; Edited 13-Apr-88 11:46 by woz
```

;;; find the first linear item starting from this point, expanding subnodes

```
(IL:|do| (COND
           ((NOT (IL:LISTP LINEAR))
            ;; we're at the end of this node's linear form -- continue from where it appeared in its super
            (IL:SETQ LINEAR (CDR (IL:|fetch| LINEAR-THREAD IL:|of| (IL:|fetch| DESTINATION IL:|of| LINEAR)))))
           ((IL:|type?| WEAK-LINK (CAR LINEAR))
            ;; it's a subnode -- examine its linear form
            (IL:SETQ LINEAR (IL:|fetch| LINEAR-FORM IL:|of| (IL:FETCH DESTINATION IL:OF (CAR LINEAR)))))
           (T (RETURN LINEAR))))))
```

(OUTPUT-BITMAP

```
(IL:LAMBDA (CONTEXT BITMAP)                                     ; Edited 17-Nov-87 11:39 by DCB
```

;;; insert a bitmap at this point in the linear form

```
(COND
   ((IL:FETCH RELINEARIZATION-TIME-STAMP IL:OF CONTEXT)
    ;; we're editing.  if this bitmap wasn't already there, insert it (this means we're no longer matching)
    (WHEN (NOT (NEXT-LINEAR CONTEXT BITMAP))
        (SET-LINEAR CONTEXT (CONS BITMAP (IL:FETCH LINEAR-POINTER IL:OF CONTEXT)))
        (WHEN (IL:FETCH MATCHING? IL:OF CONTEXT)
            (NEW-BLOCK CONTEXT)
            (IL:REPLACE MATCHING? IL:OF CONTEXT IL:WITH NIL)))
    (STEP-LINEAR CONTEXT)
    (IL:CHANGE (IL:FETCH LINE-ASCENT IL:OF (CAR (IL:FETCH CURRENT-LINE IL:OF CONTEXT)))
           (IL:IMAX IL:DATUM (IL:IDIFFERENCE (IL:BITMAPHEIGHT (CDR BITMAP))
                                  (CAR BITMAP))))
    (IL:CHANGE (IL:FETCH LINE-DESCENT IL:OF (CAR (IL:FETCH CURRENT-LINE IL:OF CONTEXT)))
           (IL:IMAX IL:DATUM (IL:IMINUS (CAR BITMAP))))
    (ADVANCE (IL:BITMAPWIDTH (CDR BITMAP))))
   (T ;; we're pretty printing.  we haven't implemented bitmaps here.  there's no real problem, but we do have to fix linear.item.width
      (IL:SHOULDNT "the pretty printer doesn't like bitmaps")))))
```

(OUTPUT-CONSTANT-STRING

```
(IL:LAMBDA (CONTEXT STRINGITEM)                                 ; Edited 7-Jul-87 12:48 by DCB
```

;; insert a fixed string in the linear form. fixed strings are previously generated stringitems (improves efficiency)

```
(COND
   ((IL:FETCH RELINEARIZATION-TIME-STAMP IL:OF CONTEXT)
    ;; we're editing.  if this stringitem wasn't already there, insert it (this means we're no longer matching)
    (COND
       ((NEXT-LINEAR CONTEXT STRINGITEM)
        (STEP-LINEAR CONTEXT))
       (T ;; this is gratuitously complicated.  it could be like output.bitmap (except that this is marginally faster)
          (LET ((LINEAR (CONS STRINGITEM (IL:FETCH LINEAR-POINTER IL:OF CONTEXT))))
```

```
                    (IF (IL:LISTP (IL:FETCH LINEAR-PREV IL:OF CONTEXT))
                        (RPLACD (IL:FETCH LINEAR-PREV IL:OF CONTEXT)
                               LINEAR)
                        (IL:REPLACE LINEAR-FORM IL:OF (IL:FETCH LINEAR-PREV IL:OF CONTEXT) IL:WITH LINEAR))
                    (IL:REPLACE LINEAR-PREV IL:OF CONTEXT IL:WITH LINEAR)
                    (WHEN (IL:FETCH MATCHING? IL:OF CONTEXT)
                        (NEW-BLOCK CONTEXT)
                        (IL:REPLACE BLOCK-START IL:OF (IL:FETCH CURRENT-BLOCK IL:OF CONTEXT) IL:WITH LINEAR)
                        (IL:REPLACE MATCHING? IL:OF CONTEXT IL:WITH NIL)))))
            (ADVANCE (IL:FETCH WIDTH IL:OF STRINGITEM))
            (IL:CHANGE (IL:FETCH LINE-ASCENT IL:OF (CAR (IL:FETCH CURRENT-LINE IL:OF CONTEXT)))
                   (IL:IMAX IL:DATUM (IL:FONTPROP (IL:FETCH FONT IL:OF STRINGITEM)
                                                  'IL:ASCENT)))
            (IL:CHANGE (IL:FETCH LINE-DESCENT IL:OF (CAR (IL:FETCH CURRENT-LINE IL:OF CONTEXT)))
                   (IL:IMAX IL:DATUM (IL:FONTPROP (IL:FETCH FONT IL:OF STRINGITEM)
                                                  'IL:DESCENT))))
           (T ;; we're pretty printing.  we have to map the font because TEDIT.INSERT does weird things with interpress fonts
              (IL:TEDIT.INSERT (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                     (COND
                        ((IL:FETCH PRIN-2? IL:OF STRINGITEM)          ; read table specific
                         (IL:MKSTRING (IL:FETCH STRING IL:OF STRINGITEM)
                                T))
                        (T (IL:FETCH STRING IL:OF STRINGITEM)))
                     NIL
                     (MAP-FONT (IL:FETCH FONT IL:OF STRINGITEM)
                           (IL:FETCH ENVIRONMENT IL:OF CONTEXT))
                     T)
              (IL:ADD (IL:FETCH CURRENT-X IL:OF CONTEXT)
                     (IL:FETCH WIDTH IL:OF STRINGITEM))))))
```

(OUTPUT-CR

```
(IL:LAMBDA (CONTEXT INDENT LINESKIP)                                    ; Edited  8-Jul-87 17:21 by DCB
    ;; insert a line start in the linear form.  this is rather tricky because we need to update the window as we go
    (COND
       ((IL:FETCH RELINEARIZATION-TIME-STAMP IL:OF CONTEXT)
        ;; we're editing.  compute various dimensions
        (LET ((LAST-LINE (CAR (IL:FETCH CURRENT-LINE IL:OF CONTEXT)))
              (CURRENT-NODE (IL:FETCH CURRENT-NODE IL:OF CONTEXT))
              Y THIS-LINE MATCH-X MATCH-BASELINE MATCH-ASCENT MATCH-DESCENT)
             (WHEN (NULL LINESKIP)
                 (IL:SETQ LINESKIP (IL:FETCH DEFAULT-LINE-SKIP IL:OF (IL:FETCH ENVIRONMENT IL:OF CONTEXT))))
             (IL:SETQ Y (IL:IDIFFERENCE (IL:FETCH YCOORD IL:OF LAST-LINE)
                               (IL:FETCH LINE-HEIGHT IL:OF LAST-LINE)))
             (IL:REPLACE ACTUAL-WIDTH IL:OF CURRENT-NODE IL:WITH (IL:IMAX (IL:FETCH ACTUAL-WIDTH IL:OF
                                                                                    CURRENT-NODE
                                                                                                )
                                                                          (IL:FETCH CURRENT-X IL:OF CONTEXT)))
            ;; if there's already a line start at this point, we can smash it, but we have to cache its old values for use when fixing up the screen
            (COND
               ((AND (IL:LISTP (IL:FETCH LINEAR-POINTER IL:OF CONTEXT))
                     (IL:TYPE? LINE-START (IL:SETQ THIS-LINE (CAR (IL:FETCH LINEAR-POINTER IL:OF CONTEXT)))))
                (WHEN (NOT (IL:FETCH BELOW? IL:OF CONTEXT))
                    (IL:REPLACE CACHED-ASCENT IL:OF THIS-LINE IL:WITH (IL:SETQ MATCH-ASCENT
                                                                              (IL:FETCH LINE-ASCENT IL:OF THIS-LINE)))
                    (IL:REPLACE CACHED-DESCENT IL:OF THIS-LINE IL:WITH (IL:SETQ MATCH-DESCENT
                                                                               (IL:FETCH LINE-DESCENT IL:OF THIS-LINE)))
                    (IL:REPLACE CACHED-Y IL:OF THIS-LINE IL:WITH (IL:SETQ MATCH-BASELINE (IL:FETCH BASE-LINE-Y
                                                                                                 IL:OF THIS-LINE)))
                    (IL:REPLACE CACHE-TIME IL:OF THIS-LINE IL:WITH (IL:FETCH RELINEARIZATION-TIME-STAMP
                                                                           IL:OF CONTEXT))
                    (IL:SETQ MATCH-X (IL:FETCH INDENT IL:OF THIS-LINE)))
                (IL:REPLACE PREV-LINE IL:OF THIS-LINE IL:WITH (IL:FETCH CURRENT-LINE IL:OF CONTEXT))
                (IL:REPLACE LINE-SKIP IL:OF THIS-LINE IL:WITH LINESKIP)
                (IL:REPLACE LINE-ASCENT IL:OF THIS-LINE IL:WITH 0)
                (IL:REPLACE LINE-DESCENT IL:OF THIS-LINE IL:WITH 0)
                (IL:REPLACE INDENT IL:OF THIS-LINE IL:WITH INDENT)
                (IL:REPLACE YCOORD IL:OF THIS-LINE IL:WITH Y))
               (T ;; there was no line start here before.  create one
                  (SET-LINEAR CONTEXT
                         (CONS (IL:SETQ THIS-LINE
                                (IL:CREATE LINE-START
                                       PREV-LINE IL:_ (IL:FETCH CURRENT-LINE IL:OF CONTEXT)
                                       NODE IL:_ (IL:FETCH CURRENT-NODE IL:OF CONTEXT)
                                       LINE-SKIP IL:_ LINESKIP
                                       LINE-ASCENT IL:_ 0
                                       LINE-DESCENT IL:_ 0
                                       INDENT IL:_ INDENT
                                       YCOORD IL:_ Y))
                               (IL:FETCH LINEAR-POINTER IL:OF CONTEXT)))
                  (WHEN (IL:FETCH MATCHING? IL:OF CONTEXT)
                      (IL:SETQ MATCH-X (IL:FETCH CURRENT-BLOCK IL:OF CONTEXT))
```

```
                    (IL:SETQ MATCH-ASCENT (IL:FETCH BLOCK-ASCENT IL:OF MATCH-X))
                    (IL:SETQ MATCH-DESCENT (IL:FETCH BLOCK-DESCENT IL:OF MATCH-X))
                    (IL:SETQ MATCH-BASELINE (IL:FETCH BLOCK-BASE-LINE IL:OF MATCH-X))
                    (IL:SETQ MATCH-X (IL:IPLUS (IL:FETCH CURRENT-X IL:OF CONTEXT)
                                               (IL:IDIFFERENCE (IL:FETCH BLOCK-X IL:OF MATCH-X)
                                                               (IL:FETCH BLOCK-NEW-X IL:OF MATCH-X)))))))
          (IL:REPLACE LINE-LENGTH IL:OF LAST-LINE IL:WITH (IL:FETCH CURRENT-X IL:OF CONTEXT))
          (IL:SELECTQ (IL:FETCH BELOW? IL:OF CONTEXT)
                (NIL                                          ; this line might be visible.  flush it and reset the block list
                     (IL:LINE-FINISHED CONTEXT (IL:FETCH CURRENT-X IL:OF CONTEXT)
                            (IL:FETCH LINEAR-POINTER IL:OF CONTEXT))
                     (IL:REPLACE SHIFT-Y IL:OF CONTEXT IL:WITH NIL)
                     (LET ((BLOCK (IL:FETCH FIRST-BLOCK IL:OF CONTEXT)))
                          (IL:REPLACE CURRENT-BLOCK IL:OF CONTEXT IL:WITH BLOCK)
                          (IL:REPLACE BLOCK-NEW-X IL:OF BLOCK IL:WITH INDENT)
                          (IL:REPLACE BLOCK-START IL:OF BLOCK IL:WITH (IL:FETCH LINEAR-POINTER IL:OF CONTEXT))
                          (WHEN (IL:REPLACE BITS? IL:OF BLOCK IL:WITH (IL:FETCH MATCHING? IL:OF CONTEXT))
                              (IL:REPLACE BLOCK-X IL:OF BLOCK IL:WITH MATCH-X)
                              (IL:REPLACE BLOCK-BASE-LINE IL:OF BLOCK IL:WITH MATCH-BASELINE)
                              (IL:REPLACE BLOCK-ASCENT IL:OF BLOCK IL:WITH MATCH-ASCENT)
                              (IL:REPLACE BLOCK-DESCENT IL:OF BLOCK IL:WITH MATCH-DESCENT))))
                (NEW                                          ; we're repainting this window from the top.  nothing should be
                                                              ; reused
                     (REPAINT-NEW-LINE (IL:FETCH CURRENT-LINE IL:OF CONTEXT)))
                NIL)
          (IL:REPLACE CURRENT-LINE IL:OF CONTEXT IL:WITH (IL:FETCH LINEAR-POINTER IL:OF CONTEXT))
          (IL:REPLACE NEXT-LINE IL:OF LAST-LINE IL:WITH (IL:FETCH CURRENT-LINE IL:OF CONTEXT))
          (IL:REPLACE CURRENT-X IL:OF CONTEXT IL:WITH INDENT)
          (STEP-LINEAR CONTEXT)))
       (T ;; we're pretty printing
          (IL:TEDIT.INSERT (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                 (IL:FCHARACTER (IL:CHARCODE IL:CR))
                 NIL NIL T)
          (IL:TEDIT.PARALOOKS (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                 (LIST '-1-STLEFTMARGIN (IL:FIXR (IL:QUOTIENT INDENT IL:MICASPERPT))
                       'PARALEADING
                       (OR LINESKIP (IL:FETCH DEFAULT-LINE-SKIP IL:OF (IL:FETCH ENVIRONMENT IL:OF CONTEXT)))))
          (IL:REPLACE CURRENT-X IL:OF CONTEXT IL:WITH INDENT)))))
```

(OUTPUT-SPACE

```
  (IL:LAMBDA (CONTEXT X)                                      ; Edited 17-Nov-87 11:40 by DCB
```

;;; insert horizontal space at this point in the linear form

```
    (COND
       ((EQ 0 X)                                              ; insert no space;  that's easy!
        NIL)
       ((IL:FETCH RELINEARIZATION-TIME-STAMP IL:OF CONTEXT)   ; we're editing
        (COND
           ((AND (IL:LISTP (IL:FETCH LINEAR-POINTER IL:OF CONTEXT))
                 (IL:SMALLP (CAR (IL:FETCH LINEAR-POINTER IL:OF CONTEXT))))
            (WHEN (AND (IL:FETCH MATCHING? IL:OF CONTEXT)
                       (IL:NEQ (CAR (IL:FETCH LINEAR-POINTER IL:OF CONTEXT))
                               X))
                (NEW-BLOCK CONTEXT)
                (IL:REPLACE MATCHING? IL:OF CONTEXT IL:WITH NIL))
            (RPLACA (IL:FETCH LINEAR-POINTER IL:OF CONTEXT)
                   X))
           (T (SET-LINEAR CONTEXT (CONS X (IL:FETCH LINEAR-POINTER IL:OF CONTEXT)))
              (WHEN (IL:FETCH MATCHING? IL:OF CONTEXT)
                  (NEW-BLOCK CONTEXT)
                  (IL:REPLACE MATCHING? IL:OF CONTEXT IL:WITH NIL))))
        (STEP-LINEAR CONTEXT)
        (ADVANCE X))
       (T ;; we're pretty printing
          (IL:FOR I IL:FROM 1 IL:TO (IL:IQUOTIENT X (IL:FETCH SPACE-WIDTH IL:OF (IL:FETCH ENVIRONMENT IL:OF CONTEXT
                                                                                          )))
             IL:DO (IL:TEDIT.INSERT (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                          " " NIL IL:DEFAULTFONT T))
          (IL:ADD (IL:FETCH CURRENT-X IL:OF CONTEXT)
                 X)))))
```

(OUTPUT-STRING

```
  (IL:LAMBDA (CONTEXT STRING PRIN-2? FONT)                    ; Edited  7-Jul-87 12:49 by DCB
                                                              ; insert a string at this point in the linear form
    (COND
       ((IL:FETCH RELINEARIZATION-TIME-STAMP IL:OF CONTEXT)   ; we're editing
        (LET (THIS-ITEM WIDTH)
             (WHEN (NULL FONT)                                ; font defaults to the DefaultFont of this environment
                 (IL:SETQ FONT (IL:FETCH DEFAULT-FONT IL:OF (IL:FETCH ENVIRONMENT IL:OF CONTEXT))))
             (COND
                ((AND (IL:LISTP (IL:FETCH LINEAR-POINTER IL:OF CONTEXT))
                      (IL:TYPE? STRING-ITEM (IL:SETQ THIS-ITEM (CAR (IL:FETCH LINEAR-POINTER IL:OF CONTEXT)))))
```

```
                                                                    ; there was already a string at this point.  is it the same one?
                (COND
                   ((OR (IL:NEQ (IL:FETCH STRING IL:OF THIS-ITEM)
                                STRING)
                        (IL:NEQ (IL:FETCH FONT IL:OF THIS-ITEM)
                                FONT)
                        (IL:NEQ (IL:FETCH PRIN-2? IL:OF THIS-ITEM)
                                PRIN-2?))                           ; it's different.  reuse the structure, but recompute everything and
                                                                    ; smash all the fields
                                                                    ; read table specific
                    (IL:SETQ WIDTH (STRINGWIDTH STRING FONT PRIN-2?))
                    (IL:REPLACE STRING IL:OF THIS-ITEM IL:WITH STRING)
                    (IL:REPLACE WIDTH IL:OF THIS-ITEM IL:WITH WIDTH)
                    (IL:REPLACE FONT IL:OF THIS-ITEM IL:WITH FONT)
                    (IL:REPLACE PRIN-2? IL:OF THIS-ITEM IL:WITH PRIN-2?)
                    (WHEN (IL:FETCH MATCHING? IL:OF CONTEXT)
                        (NEW-BLOCK CONTEXT)
                        (IL:REPLACE MATCHING? IL:OF CONTEXT IL:WITH NIL)))
                   (T                                               ; it's the same.  this is easy
                      (IL:SETQ WIDTH (IL:FETCH WIDTH IL:OF THIS-ITEM)))))
               (T                                                   ; we need to create a new StringItem
                                                                    ; read table specific
                  (IL:SETQ WIDTH (STRINGWIDTH STRING FONT PRIN-2?))
                  (SET-LINEAR CONTEXT
                         (CONS (IL:CREATE STRING-ITEM
                                          STRING IL:_ STRING
                                          WIDTH IL:_ WIDTH
                                          FONT IL:_ FONT
                                          PRIN-2? IL:_ PRIN-2?)
                               (IL:FETCH LINEAR-POINTER IL:OF CONTEXT)))
                  (WHEN (IL:FETCH MATCHING? IL:OF CONTEXT)          ; not anymore
                      (NEW-BLOCK CONTEXT)
                      (IL:REPLACE MATCHING? IL:OF CONTEXT IL:WITH NIL))))
            (STEP-LINEAR CONTEXT)
            (IL:CHANGE (IL:FETCH LINE-ASCENT IL:OF (CAR (IL:FETCH CURRENT-LINE IL:OF CONTEXT)))
                   (IL:IMAX IL:DATUM (IL:FONTPROP FONT 'IL:ASCENT)))
            (IL:CHANGE (IL:FETCH LINE-DESCENT IL:OF (CAR (IL:FETCH CURRENT-LINE IL:OF CONTEXT)))
                   (IL:IMAX IL:DATUM (IL:FONTPROP FONT 'IL:DESCENT)))
            (ADVANCE WIDTH)))
         (T ;; we're pretty printing.  we have to map the font because TEDIT.INSERT does weird things with interpress fonts
            (IL:TEDIT.INSERT (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                   (COND
                      (PRIN-2?                                      ; read table specific
                             (IL:SETQ STRING (IL:MKSTRING STRING T)))
                      (T STRING))
                   NIL
                   (MAP-FONT (OR FONT (IL:SETQ FONT (IL:FETCH DEFAULT-FONT IL:OF (IL:FETCH ENVIRONMENT IL:OF CONTEXT
                                                                                          ))))
                          (IL:FETCH ENVIRONMENT IL:OF CONTEXT))
                   T)
            (IL:ADD (IL:FETCH CURRENT-X IL:OF CONTEXT (STRINGWIDTH STRING FONT)))))))
```

(PAINT-TO-END-OF-LINE

```
  (IL:LAMBDA (CONTEXT LINEAR-END)                                   ; Edited 17-Nov-87 11:40 by DCB
```

;;; update the window to the end of the current line

```
    (LET ((THIS-LINE (CAR (IL:FETCH CURRENT-LINE IL:OF CONTEXT))))
         (COND
            ((EQ (IL:FETCH REPAINT-LINE IL:OF CONTEXT)
                 THIS-LINE)
             ;; we've already started displaying some of this line
             (LET ((BLANK-FROM (COND
                                  ((EQ (IL:FETCH REPAINT-X IL:OF CONTEXT)
                                       (IL:FETCH INDENT IL:OF THIS-LINE))
                                   ;; painting from the start of the line, so blank from left edge of window
                                   (IL:FETCH WINDOW-LEFT IL:OF CONTEXT))
                                  (T ;; just blank the part we're repainting
                                     (IL:FETCH REPAINT-X IL:OF CONTEXT)))))
                   (IL:BLTSHADE IL:WHITESHADE (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                          BLANK-FROM
                          (IL:ADD1 (IL:FETCH NEXT-LINE-Y IL:OF THIS-LINE))
                          (IL:ADD1 (IL:IDIFFERENCE (IL:FETCH WINDOW-RIGHT IL:OF CONTEXT)
                                                   BLANK-FROM))
                          (IL:FETCH LINE-HEIGHT IL:OF THIS-LINE)))
             (REPAINT CONTEXT (IL:FETCH REPAINT-X IL:OF CONTEXT)
                    (IL:FETCH BASE-LINE-Y IL:OF THIS-LINE)
                    (IL:FETCH REPAINT-START IL:OF CONTEXT)
                    LINEAR-END))
            (T ;; there are several lines which need to be repainted
               (IL:BLTSHADE IL:WHITESHADE (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
```

```
                    (IL:FETCH WINDOW-LEFT IL:OF CONTEXT)
                    (IL:ADD1 (IL:FETCH NEXT-LINE-Y IL:OF THIS-LINE))
                    (IL:ADD1 (IL:IDIFFERENCE (IL:FETCH WINDOW-RIGHT IL:OF CONTEXT)
                                             (IL:FETCH WINDOW-LEFT IL:OF CONTEXT)))
                    (IL:IDIFFERENCE (IL:FETCH YCOORD IL:OF (IL:FETCH REPAINT-LINE IL:OF CONTEXT))
                                    (IL:FETCH NEXT-LINE-Y IL:OF THIS-LINE)))
              (REPAINT CONTEXT (IL:FETCH REPAINT-X IL:OF CONTEXT)
                    (IL:FETCH BASE-LINE-Y IL:OF (IL:FETCH REPAINT-LINE IL:OF CONTEXT))
                    (CDR (IL:FETCH REPAINT-START IL:OF CONTEXT))
                    LINEAR-END))))))
```

(RECOMPUTE-FORMAT-VALUES

```
(IL:LAMBDA (NODE CONTEXT)                                   ; Edited 7-Jul-87 12:49 by DCB
  (LET (CHANGED?)
       (IF (EQ (IL:FETCH NODE-TYPE IL:OF NODE)
               TYPE-LITATOM)
           (IL:SETQ CHANGED? (IL:FETCH INLINE-WIDTH IL:OF NODE))
           (IL:FOR SUBNODE IL:IN (CDR (IL:FETCH SUB-NODES IL:OF NODE)) IL:DO (RECOMPUTE-FORMAT-VALUES SUBNODE
                                                                                   CONTEXT)
                                                                          (WHEN (IL:FETCH CHANGED? IL:OF
                                                                                                   SUBNODE
                                                                                                   )
                                                                                (IL:SETQ CHANGED? T))))
       (FUNCALL (IL:FETCH COMPUTE-FORMAT-VALUES IL:OF (IL:FETCH NODE-TYPE IL:OF NODE))
              NODE
              (IL:FETCH ENVIRONMENT IL:OF CONTEXT)
              CONTEXT)
       (WHEN (AND CHANGED? (IL:NEQ CHANGED? (IL:FETCH INLINE-WIDTH IL:OF NODE)))
             (IL:REPLACE CHANGED? IL:OF NODE IL:WITH T)))))
```

(RELINEARIZE

```
(IL:LAMBDA (NODE CONTEXT)                                   ; Edited 7-Apr-88 11:04 by woz
```

;;; some part of this node has changed. do all the work necessary to update the linear form and window. this function is never supposed to be an entry ;;; point. that is, it assumes it will run under an sedit profile.

```
    (LET ((SUPER-NODE (IL:|fetch| SUPER-NODE IL:|of| NODE))
          FOLLOWING-LINE FROM-TOP (OLD-ACTUAL-WIDTH (IL:|fetch| ACTUAL-WIDTH IL:|of| NODE))
          (OLD-ACTUALLLENGTH (IL:|fetch| ACTUAL-LLENGTH IL:|of| NODE))
          (OLD-LAST-LINE (IL:|fetch| LAST-LINE IL:|of| NODE))
          (DISPLAY-WINDOW-REGION (IL:DSPCLIPPINGREGION NIL (IL:|fetch| DISPLAY-WINDOW IL:|of| CONTEXT)))
          Y-1 Y-2)                                          ; we cache the window dimensions because they're needed so
                                                            ; often
         (IL:|replace| WINDOW-LEFT IL:|of| CONTEXT IL:|with| (IL:|fetch| (IL:REGION IL:LEFT) IL:|of| DISPLAY-WINDOW-REGION))
         (IL:|replace| WINDOW-BOTTOM IL:|of| CONTEXT IL:|with| (IL:|fetch| (IL:REGION IL:BOTTOM) IL:|of| DISPLAY-WINDOW-REGION
                                                               ))
         (IL:|replace| WINDOW-RIGHT IL:|of| CONTEXT IL:|with| (IL:|fetch| (IL:REGION IL:RIGHT) IL:|of| DISPLAY-WINDOW-REGION))
         (IL:|replace| WINDOW-TOP IL:|of| CONTEXT IL:|with| (IL:|fetch| (IL:REGION IL:TOP) IL:|of| DISPLAY-WINDOW-REGION))
         (COND
            (SUPER-NODE                                     ; the usual case: some node changed and we want to do a
                                                            ; minimal update
                    (IL:|replace| RELINEARIZATION-TIME-STAMP IL:|of| CONTEXT IL:|with| (IL:ADD1 (IL:|fetch|
                                                                                                 RELINEARIZATION-TIME-STAMP
                                                                                                    IL:|of| CONTEXT)))
                    (IL:|replace| SHIFT-Y IL:|of| CONTEXT IL:|with| NIL)
                    (IL:|replace| SHIFT-DOWN IL:|of| CONTEXT IL:|with| 0)
                    (LET ((FIRST-LINE (IL:|fetch| FIRST-LINE IL:|of| NODE))
                          FIRST-LINE-LINEAR)
                         (IL:SETQ FIRST-LINE-LINEAR (FIRST-LINE-LINEAR NODE CONTEXT))
                         (COND
                            ((IL:|replace| BELOW? IL:|of| CONTEXT IL:|with| (IL:ILESSP (IL:|fetch| YCOORD
                                                                                             IL:|of| (IL:|fetch| FIRST-LINE
                                                                                                        IL:|of| NODE))
                                                                                      (IL:|fetch| WINDOW-BOTTOM IL:|of| CONTEXT)))
                             (IL:|replace| MATCHING? IL:|of| CONTEXT IL:|with| NIL))
                            (T (IL:|replace| MATCHING? IL:|of| CONTEXT IL:|with| (IL:NEQ (IL:|fetch| START-X IL:|of| NODE)
                                                                                         0))
                               (LET ((BLOCK (IL:|fetch| FIRST-BLOCK IL:|of| CONTEXT)))
                                    (IL:|replace| CURRENT-BLOCK IL:|of| CONTEXT IL:|with| BLOCK)
                                    (IL:|replace| VISIBLE? IL:|of| CONTEXT IL:|with| (IL:ILESSP (IL:|fetch| NEXT-LINE-Y
                                                                                                      IL:|of| FIRST-LINE)
                                                                                               (IL:|fetch| WINDOW-TOP
                                                                                                  IL:|of| CONTEXT)))
                                    (IL:|replace| BLOCK-START IL:|of| BLOCK IL:|with| FIRST-LINE-LINEAR)
                                    (IL:|replace| BLOCK-X IL:|of| BLOCK IL:|with| (IL:|fetch| INDENT IL:|of| FIRST-LINE))
                                    (IL:|replace| BLOCK-NEW-X IL:|of| BLOCK IL:|with| (IL:|fetch| INDENT IL:|of| FIRST-LINE))
                                    (COND
                                       ((IL:|fetch| MATCHING? IL:|of| CONTEXT)
                                        (IL:|replace| BITS? IL:|of| BLOCK IL:|with| T)
                                        (IL:|replace| BLOCK-WIDTH IL:|of| BLOCK IL:|with| (IL:IDIFFERENCE (IL:|fetch|
                                                                                                              CURRENT-X
                                                                                                           IL:|of| CONTEXT
                                                                                                           )
                                                                                                  (IL:|fetch| BLOCK-X
                                                                                                     IL:|of| BLOCK)))
```

```
                                        (IL:|replace| BLOCK-BASE-LINE IL:|of| BLOCK IL:|with| (IL:|fetch| BASE-LINE-Y
                                                                                                IL:|of| FIRST-LINE))
                                        (IL:|replace| BLOCK-ASCENT IL:|of| BLOCK IL:|with| (IL:|fetch| LINE-ASCENT
                                                                                            IL:|of| FIRST-LINE))
                                        (IL:|replace| BLOCK-DESCENT IL:|of| BLOCK IL:|with| (IL:|fetch| LINE-DESCENT
                                                                                              IL:|of| FIRST-LINE)))
                                     (T (IL:|replace| BITS? IL:|of| BLOCK IL:|with| NIL)))
                                  (IL:|replace| REPAINT-X IL:|of| CONTEXT IL:|with| (IL:|fetch| INDENT IL:|of| FIRST-LINE))
                                  (IL:|replace| REPAINT-LINE IL:|of| CONTEXT IL:|with| FIRST-LINE)
                                  (IL:|replace| REPAINT-START IL:|of| CONTEXT IL:|with| FIRST-LINE-LINEAR))))
                        (IL:|replace| CACHED-ASCENT IL:|of| FIRST-LINE IL:|with| (IL:|fetch| LINE-ASCENT IL:|of| FIRST-LINE))
                        (IL:|replace| CACHED-DESCENT IL:|of| FIRST-LINE IL:|with| (IL:|fetch| LINE-DESCENT IL:|of| FIRST-LINE)
                              )
                        (IL:|replace| CACHED-Y IL:|of| FIRST-LINE IL:|with| (IL:|fetch| BASE-LINE-Y IL:|of| FIRST-LINE))
                        (IL:|replace| CACHE-TIME IL:|of| FIRST-LINE IL:|with| (IL:|fetch| RELINEARIZATION-TIME-STAMP
                                                                                 IL:|of| CONTEXT))
                        (SCAN-FOR-BOUNDS (CDR FIRST-LINE-LINEAR)
                               (IL:|fetch| LINEAR-THREAD IL:|of| NODE)
                               FIRST-LINE-LINEAR T)))
    (T ;; we're redisplaying everything from scratch (probably because the window was reshaped).  node is the root node
       (IL:BLTSHADE IL:WHITESHADE (IL:|fetch| DISPLAY-WINDOW IL:|of| CONTEXT)
              (IL:|fetch| (IL:REGION IL:LEFT) IL:|of| DISPLAY-WINDOW-REGION)
              (IL:|fetch| (IL:REGION IL:BOTTOM) IL:|of| DISPLAY-WINDOW-REGION)
              (IL:|fetch| (IL:REGION IL:WIDTH) IL:|of| DISPLAY-WINDOW-REGION)
              (IL:|fetch| (IL:REGION IL:HEIGHT) IL:|of| DISPLAY-WINDOW-REGION))
       (IL:|replace| SHIFT-DOWN IL:|of| CONTEXT IL:|with| NIL)
       (IL:SETQ FROM-TOP T)
       (IL:|replace| BELOW? IL:|of| CONTEXT IL:|with| 'NEW)
       (IL:|replace| MATCHING? IL:|of| CONTEXT IL:|with| NIL)
       ;; must set ascent and descent of first line because the linearizer never touches this line start
       (IL:|replace| LINE-ASCENT IL:|of| (IL:|fetch| FIRST-LINE IL:|of| NODE) IL:|with| 0)
       (IL:|replace| LINE-DESCENT IL:|of| (IL:|fetch| FIRST-LINE IL:|of| NODE) IL:|with| 0)
       ;; not sure if the format values will be taken care of elsewhere, so do it here just to be sure.
       (COMPUTE-ALL-FORMATS CONTEXT)))
(WHEN (EQ (IL:|fetch| START-X IL:|of| NODE)
          0)
    (IL:SHOULDNT "the linearize root method should take care of this")
    (IL:|replace| START-X IL:|of| NODE IL:|with| (IL:|fetch| START-X IL:|of| SUPER-NODE))
    (IL:|replace| FIRST-LINE IL:|of| NODE IL:|with| (IL:|fetch| FIRST-LINE IL:|of| SUPER-NODE))
    (IL:|replace| RIGHT-MARGIN IL:|of| NODE IL:|with| (IL:|fetch| (IL:REGION IL:WIDTH) IL:|of| DISPLAY-WINDOW-REGION)
          ))
(IL:|replace| CURRENT-X IL:|of| CONTEXT IL:|with| (IL:|fetch| START-X IL:|of| NODE))
(IL:|replace| CURRENT-NODE IL:|of| CONTEXT IL:|with| SUPER-NODE)
(IL:|replace| CURRENT-LINE IL:|of| CONTEXT IL:|with| (FIRST-LINE-LINEAR NODE CONTEXT))
(IL:|replace| LINEAR-POINTER IL:|of| CONTEXT IL:|with| (IL:|fetch| LINEAR-THREAD IL:|of| NODE))
(IL:|replace| LINEAR-PREV IL:|of| CONTEXT IL:|with| NIL)
(GENERATE-LINEAR-FORM NODE CONTEXT (IF FROM-TOP
                                       (IL:|fetch| (IL:REGION IL:WIDTH) IL:|of| DISPLAY-WINDOW-REGION)
                                       (IL:|fetch| RIGHT-MARGIN IL:|of| NODE)))
;; if this isn't the top of the tree, and reformatting this node caused the width of its last line to change, the formatting of its supernode might
;; change, so we'll have to relinearize it.  and so on...
(IL:|while| (AND SUPER-NODE (IL:|fetch| SUPER-NODE IL:|of| SUPER-NODE)
                 (IL:NEQ (IL:|fetch| CURRENT-X IL:|of| CONTEXT)
                         (IL:IPLUS (IL:|fetch| START-X IL:|of| NODE)
                                   OLD-ACTUALLLENGTH)))
   IL:|do| (IL:SETQ OLD-ACTUALLLENGTH (IL:|fetch| ACTUAL-LLENGTH IL:|of| SUPER-NODE))
           (IL:SETQ OLD-LAST-LINE (IL:|fetch| LAST-LINE IL:|of| SUPER-NODE))
           (IL:|replace| LINEAR-PREV IL:|of| CONTEXT IL:|with| (IL:|fetch| LINEAR-THREAD IL:|of| NODE))
           (IL:|replace| LINEAR-POINTER IL:|of| CONTEXT IL:|with| (CDR (IL:|fetch| LINEAR-PREV IL:|of| CONTEXT)))
           (IL:|replace| LAST-LINEARIZED-SUB-NODE-INDEX IL:|of| CONTEXT IL:|with| (IL:|fetch| SUB-NODE-INDEX
                                                                                    IL:|of| NODE))
           ;; compute the maximum width of the lines in the linear form of the super up to the end of the node we just linearized (so we can
           ;; recompute the super's width)
           (IL:|replace| ACTUAL-WIDTH IL:|of| SUPER-NODE
              IL:|with| (IL:|bind| (WIDTH IL:_ 0) IL:|for| (LINE IL:_ (IL:|fetch| FIRST-LINE IL:|of| SUPER-NODE))
                        IL:|by| (CAR (IL:|fetch| NEXT-LINE IL:|of| LINE))
                        IL:|while| (IL:NEQ LINE (CAR (IL:|fetch| CURRENT-LINE IL:|of| CONTEXT)))
                        IL:|do| (WHEN (IL:IGREATERP (IL:|fetch| LINE-LENGTH IL:|of| LINE)
                                                     WIDTH)
                                      (IL:SETQ WIDTH (IL:|fetch| LINE-LENGTH IL:|of| LINE)))
                        IL:|finally| (RETURN WIDTH)))
           (IL:|replace| ACTUAL-LLENGTH IL:|of| SUPER-NODE IL:|with| NIL)
           (FUNCALL (IL:|fetch| LINEARIZE IL:|of| (IL:|fetch| NODE-TYPE IL:|of| SUPER-NODE))
                  SUPER-NODE CONTEXT (IL:|fetch| SUB-NODE-INDEX IL:|of| NODE))
           (WHEN (NOT (AND (IL:|type?| WEAK-LINK (IL:|fetch| LINEAR-POINTER IL:|of| CONTEXT))
                           (EQ (IL:FETCH DESTINATION IL:OF (IL:|fetch| LINEAR-POINTER IL:|of| CONTEXT))
                               SUPER-NODE)))
               (SET-LINEAR CONTEXT (CDR (LAST (IL:FETCH LINEAR-FORM IL:OF SUPER-NODE)))))
           ;; this used to be:
           ;; (il:|replace| last-line-linear il:|of| super-node il:|with| (il:fetch current-line il:of context))
           (IL:|replace| LAST-LINE IL:|of| SUPER-NODE IL:|with| (CAR (IL:|fetch| CURRENT-LINE IL:|of| CONTEXT)))
```

```
                (IL:|replace| ACTUAL-WIDTH IL:|of| SUPER-NODE IL:|with| (IL:IDIFFERENCE (IL:IMAX (IL:|fetch| ACTUAL-WIDTH
                                                                                                  IL:|of| SUPER-NODE)
                                                                                           (IL:|fetch| CURRENT-X
                                                                                              IL:|of| CONTEXT))
                                                                                 (IL:|fetch| START-X IL:|of| SUPER-NODE)))
                (IL:|replace| ACTUAL-LLENGTH IL:|of| SUPER-NODE IL:|with| (IL:IDIFFERENCE (IL:|fetch| CURRENT-X
                                                                                              IL:|of| CONTEXT)
                                                                                       (IL:|fetch| START-X IL:|of| SUPER-NODE)))
                (IL:|replace| CHANGED? IL:|of| SUPER-NODE IL:|with| NIL)
                (IL:SETQ NODE SUPER-NODE)
                (IL:SETQ SUPER-NODE (IL:|fetch| SUPER-NODE IL:|of| SUPER-NODE))
                (IL:|replace| CURRENT-NODE IL:|of| CONTEXT IL:|with| SUPER-NODE))
        (IL:|replace| LINE-LENGTH IL:|of| (IL:|fetch| LAST-LINE IL:|of| NODE) IL:|with| (IL:|fetch| CURRENT-X IL:|of| CONTEXT))
        (COND
           ((OR (NULL SUPER-NODE)
                (NULL (IL:|fetch| SUPER-NODE IL:|of| SUPER-NODE)))
            ;; need to fix up node and supernode pointers, because came through root path
            (WHEN (NULL SUPER-NODE)
                (IL:SETQ SUPER-NODE NODE)
                (IL:SETQ NODE (SUBNODE 1 NODE)))
            ;; we've relinearized to the end of the structure, so all we need to do is make sure the last line is flushed, blank the rest of the window,
            ;; and fix up some recorded dimensions
            (IL:|replace| ACTUAL-LLENGTH IL:|of| SUPER-NODE IL:|with| (IL:|fetch| ACTUAL-LLENGTH IL:|of| NODE))
            (IL:|replace| NEXT-LINE IL:|of| (IL:|fetch| LAST-LINE IL:|of| NODE) IL:|with| NIL)
            (IL:SELECTQ (IL:|fetch| BELOW? IL:|of| CONTEXT)
                  (T)
                  (NIL (LINE-FINISHED CONTEXT (IL:|fetch| CURRENT-X IL:|of| CONTEXT)
                              NIL T))
                  (NEW (REPAINT-NEW-LINE (LAST-LINE-LINEAR NODE CONTEXT)))
                  (IL:SHOULDNT "unexpected value for Below?"))
            (LET* ((BOTTOM-Y (IL:ADD1 (IL:|fetch| NEXT-LINE-Y IL:|of| (IL:|fetch| LAST-LINE IL:|of| NODE))))
                   (EXTENT (IL:WINDOWPROP (IL:|fetch| DISPLAY-WINDOW IL:|of| CONTEXT)
                                'IL:EXTENT))
                   (OLD-BOTTOM-Y (IL:IDIFFERENCE (IL:|fetch| (IL:REGION IL:BOTTOM) IL:|of| EXTENT)
                                       (OR (IL:|fetch| SHIFT-DOWN IL:|of| CONTEXT)
                                           0))))
                  (WHEN (AND (IL:NEQ (IL:|fetch| BELOW? IL:|of| CONTEXT)
                                     T)
                             (IL:IGREATERP BOTTOM-Y OLD-BOTTOM-Y))
                      (IL:BLTSHADE IL:WHITESHADE (IL:|fetch| DISPLAY-WINDOW IL:|of| CONTEXT)
                             (IL:|fetch| (IL:REGION IL:LEFT) IL:|of| DISPLAY-WINDOW-REGION)
                             OLD-BOTTOM-Y
                             (IL:|fetch| (IL:REGION IL:WIDTH) IL:|of| DISPLAY-WINDOW-REGION)
                             (IL:IDIFFERENCE BOTTOM-Y OLD-BOTTOM-Y)))
                  (IL:|replace| (IL:REGION IL:BOTTOM) IL:|of| EXTENT IL:|with| BOTTOM-Y)
                  (IL:|replace| (IL:REGION IL:HEIGHT) IL:|of| EXTENT IL:|with| (IL:IDIFFERENCE 1 BOTTOM-Y))))
           (T ;; we've finished relinearizing, but there was stuff after this.  patch the pieces together and fix up all sorts of things
              (IL:|add| (IL:|fetch| LINE-LENGTH IL:|of| (IL:|fetch| LAST-LINE IL:|of| NODE))
                     (SCAN-FOR-BOUNDS (CDR (IL:|fetch| LINEAR-THREAD IL:|of| NODE))
                            NIL
                            (LAST-LINE-LINEAR NODE CONTEXT)))
              (IL:SETQ FOLLOWING-LINE (CAR (IL:|fetch| NEXT-LINE IL:|of| (IL:|fetch| LAST-LINE IL:|of| NODE))))
              (WHEN (NOT (IL:|fetch| BELOW? IL:|of| CONTEXT))
                  (NEW-BLOCK CONTEXT)
                  (LET ((BLOCK (IL:|fetch| CURRENT-BLOCK IL:|of| CONTEXT)))
                       (IL:|replace| BLOCK-START IL:|of| BLOCK IL:|with| (CDR (IL:|fetch| LINEAR-THREAD IL:|of| NODE)))
                       (IL:|replace| BLOCK-NEW-X IL:|of| BLOCK IL:|with| (IL:|fetch| CURRENT-X IL:|of| CONTEXT))
                       (IL:|replace| BLOCK-X IL:|of| BLOCK IL:|with| (IL:|fetch| CURRENT-X IL:|of| CONTEXT))
                       (IL:|replace| BITS? IL:|of| BLOCK IL:|with| T)
                       (COND
                          ((EQ (IL:|fetch| CACHE-TIME IL:|of| OLD-LAST-LINE)
                               (IL:|fetch| RELINEARIZATION-TIME-STAMP IL:|of| CONTEXT))
                           (IL:|replace| BLOCK-BASE-LINE IL:|of| BLOCK IL:|with| (IL:|fetch| CACHED-Y IL:|of| OLD-LAST-LINE))
                           (IL:|replace| BLOCK-ASCENT IL:|of| BLOCK IL:|with| (IL:|fetch| CACHED-ASCENT IL:|of| OLD-LAST-LINE
                                                                                          ))
                           (IL:|replace| BLOCK-DESCENT IL:|of| BLOCK IL:|with| (IL:|fetch| CACHED-DESCENT IL:|of|
                                                                                                  OLD-LAST-LINE
                                                                                                  )))
                          (T (IL:|replace| BLOCK-BASE-LINE IL:|of| BLOCK IL:|with| (IL:|fetch| BASE-LINE-Y IL:|of|
                                                                                                  OLD-LAST-LINE
                                                                                                  ))
                             (IL:|replace| BLOCK-ASCENT IL:|of| BLOCK IL:|with| (IL:|fetch| LINE-ASCENT IL:|of| OLD-LAST-LINE
                                                                                          ))
                             (IL:|replace| BLOCK-DESCENT IL:|of| BLOCK IL:|with| (IL:|fetch| LINE-DESCENT IL:|of|
                                                                                                  OLD-LAST-LINE
                                                                                                  ))))
                       (IL:SETQ Y-1 (IL:IDIFFERENCE (IL:|fetch| BLOCK-BASE-LINE IL:|of| BLOCK)
                                          (IL:IPLUS (IL:|fetch| BLOCK-DESCENT IL:|of| BLOCK)
                                                 1)))
                       (IL:SETQ Y-2 (IL:|fetch| NEXT-LINE-Y IL:|of| (CAR (IL:|fetch| CURRENT-LINE IL:|of| CONTEXT))))))
                  (LINE-FINISHED CONTEXT (IL:|fetch| LINE-LENGTH IL:|of| (IL:|fetch| LAST-LINE IL:|of| NODE))
                         (IL:|fetch| NEXT-LINE IL:|of| (IL:|fetch| LAST-LINE IL:|of| NODE))
                         T)
                  (IL:SETQ Y-1 (IL:IDIFFERENCE Y-1 (IL:|fetch| SHIFT-DOWN IL:|of| CONTEXT))))))
```

```
            (CLEAN-UP-AFTER-RELINEARIZATION CONTEXT NODE FOLLOWING-LINE Y-1 Y-2)))
                                                    ; changing this node may have changed the width of some of its
                                                    ; super nodes
        (PROPAGATE-WIDTH-CHANGE CONTEXT NODE OLD-ACTUAL-WIDTH))))
```

(REPAINT

```
(IL:LAMBDA (CONTEXT X Y LINEAR-START END)                    ; Edited 11-Apr-88 15:52 by woz
```

;; display the sequence of linear form from linear.start to end, starting at x,y. end is either an integer, indicating the lowest y to which repainting ;; should be done, or a linear form pointer

```
(IL:|bind| (DSP IL:_ (IL:|fetch| DISPLAY-WINDOW IL:|of| CONTEXT))
       ITEM TEMP MIN-Y CURRENT-FONT IL:|first| (IL:MOVETO X Y DSP)
                                    (COND
                                       ((IL:FIXP END)
                                        (IL:SETQ MIN-Y END)
                                        (IL:SETQ END NIL))
                                       (T (IL:SETQ MIN-Y (IL:|fetch| WINDOW-BOTTOM IL:|of| CONTEXT))))
    IL:|while| (IL:NEQ LINEAR-START END) IL:|do| (COND
                                       ((NOT (IL:LISTP LINEAR-START))
                                                    ; finished this node, follow its thread to super
                                        (IL:SETQ LINEAR-START (CDR (IL:|fetch| LINEAR-THREAD
                                                                               IL:|of| (IL:|fetch| DESTINATION
                                                                                                   IL:|of| LINEAR-START)))))
                                       ((IL:|type?| WEAK-LINK (IL:SETQ ITEM (CAR LINEAR-START)))
                                                    ; insert the linear form of a subnode
                                        (IL:SETQ LINEAR-START (IL:|fetch| LINEAR-FORM
                                                                          IL:|of| (IL:|fetch| DESTINATION
                                                                                              IL:|of| ITEM))))
                                       (T           ; display something
                                          (COND
                                             ((IL:|type?| LINE-START ITEM)
                                                    ; new line.  if it takes us off the bottom of the region to be
                                                    ; repainted, we can quit
                                              (WHEN (IL:ILEQ (IL:IPLUS (IL:SETQ Y
                                                                                (IL:|fetch| BASE-LINE-Y
                                                                                            IL:|of| ITEM))
                                                                       (IL:|fetch| LINE-ASCENT IL:|of| ITEM))
                                                             MIN-Y)
                                                    ; we've repainted enough
                                                    (RETURN))
                                              (IL:MOVETO (IL:|fetch| INDENT IL:|of| ITEM)
                                                     Y DSP))
                                             ((IL:FIXP ITEM)
                                              (IL:RELMOVETO ITEM 0 DSP))
                                             ((IL:|type?| STRING-ITEM ITEM)
                                              (WHEN (IL:NEQ CURRENT-FONT (IL:|fetch| FONT IL:|of| ITEM))
                                                    (WHEN (NULL (IL:|fetch| FONT IL:|of| ITEM))
                                                          (IL:SHOULDNT "this StringItem has no font"))
                                                    (IL:DSPFONT (IL:|fetch| FONT IL:|of| ITEM)
                                                           DSP)
                                                    (IL:SETQ CURRENT-FONT (IL:|fetch| FONT IL:|of| ITEM)))
                                              (COND
                                                 ((IL:STRINGP (IL:SETQ TEMP (IL:|fetch| STRING
                                                                                        IL:|of| ITEM)))
                                                    ; read table specific
                                                  (PRINT-STRING TEMP DSP (IL:|fetch| PRIN-2?
                                                                                     IL:|of| ITEM)))
                                                 ((IL:|fetch| PRIN-2? IL:|of| ITEM)
                                                    ; read table specific
                                                  (IL:PRIN2 TEMP DSP))
                                                 (T (IL:PRIN1 TEMP DSP))))
                                             ((IL:LISTP ITEM)
                                              (IL:BITBLT (CDR ITEM)
                                                     NIL NIL DSP (IL:DSPXPOSITION NIL DSP)
                                                     (IL:IDIFFERENCE Y (CAR ITEM)))
                                              (IL:RELMOVETO (IL:BITMAPWIDTH (CDR ITEM))
                                                     0 DSP))
                                             (T (IL:SHOULDNT "unknown linear form item")))
                                          (IL:SETQ LINEAR-START (CDR LINEAR-START)))))))
```

(REUSE-LINEAR-FORM

```
(IL:LAMBDA (NODE CONTEXT)                                    ; Edited  8-Apr-88 12:06 by woz
```

;;; we've been asked to generate the linear form of node, and have decided that the old one will do. make any necessary adjustments and make sure ;;; that it's displayed properly

```
    (LET ((CURRENT-X (IL:|fetch| CURRENT-X IL:|of| CONTEXT))
          (CURRENT-LINE (IL:|fetch| CURRENT-LINE IL:|of| CONTEXT))
          DELTA-X TEMP)
         (COND
            ((IL:NEQ (IL:|fetch| START-X IL:|of| NODE)
                     0)                                      ; adjust the StartX values for this node and all its subnodes
             (WHEN (IL:NEQ (IL:SETQ DELTA-X (IL:IDIFFERENCE (IL:|fetch| CURRENT-X IL:|of| CONTEXT)
                                                            (IL:|fetch| START-X IL:|of| NODE)))
                           0)
```

```
                (SHIFT-LINEAR-FORM NODE DELTA-X)))
      (T                                                    ; this must be a prelinearized atom
         (IL:|replace| START-X IL:|of| NODE IL:|with| CURRENT-X)))
   (COND
      ((OR (NULL (IL:|fetch| LINEARIZE IL:|of| (IL:|fetch| NODE-TYPE IL:|of| NODE)))
           (IL:|fetch| INLINE? IL:|of| NODE))
       (IL:|replace| FIRST-LINE IL:|of| NODE IL:|with| (CAR CURRENT-LINE))
       (WHEN (AND (IL:|fetch| MATCHING? IL:|of| CONTEXT)
                  (IL:|fetch| CHANGED? IL:|of| NODE)
                  (NOT (IL:|fetch| LINEARIZE IL:|of| (IL:|fetch| NODE-TYPE IL:|of| NODE))))
           (NEW-BLOCK CONTEXT)
           (IL:|replace| MATCHING? IL:|of| CONTEXT IL:|with| NIL))
       (IL:SETQ CURRENT-X (IL:IPLUS CURRENT-X (SCAN-FOR-BOUNDS (IL:|fetch| LINEAR-FORM IL:|of| NODE)
                                                               (CDR (LAST (IL:|fetch| LINEAR-FORM IL:|of| NODE)))
                                                               CURRENT-LINE NIL))))
      (T                                                    ; the linear form spans several lines
         (IL:SETQ CURRENT-X (IL:IPLUS CURRENT-X (SCAN-FOR-BOUNDS (IL:|fetch| LINEAR-FORM IL:|of| NODE)
                                                                 (CDR (LAST (IL:|fetch| LINEAR-FORM IL:|of| NODE)))
                                                                 CURRENT-LINE NIL)))
         (IL:|replace| FIRST-LINE IL:|of| NODE IL:|with| (CAR CURRENT-LINE))
         (IL:SETQ TEMP (IL:|fetch| NEXT-LINE IL:|of| (CAR CURRENT-LINE)))
         (IL:|replace| PREV-LINE IL:|of| (CAR TEMP) IL:|with| CURRENT-LINE)
         (IL:|replace| LINE-LENGTH IL:|of| (CAR CURRENT-LINE) IL:|with| CURRENT-X)
         (IL:|add| (IL:|fetch| INDENT IL:|of| (CAR TEMP))
                DELTA-X)
         ;; for each line in the linear form, adjust its y coordinate and indentation and the flush it (except the last)
         (IL:|bind| (BLOCK IL:_ (IL:|fetch| FIRST-BLOCK IL:|of| CONTEXT))
                (DELTA-Y IL:_ (IL:IDIFFERENCE (IL:|fetch| NEXT-LINE-Y IL:|of| (CAR CURRENT-LINE))
                                              (IL:|fetch| YCOORD IL:|of| (CAR TEMP))))
                (BELOW? IL:_ (IL:|fetch| BELOW? IL:|of| CONTEXT))
            IL:|first| (WHEN (NOT BELOW?)
                             (LINE-FINISHED CONTEXT CURRENT-X TEMP)
                             (IL:SETQ BELOW? (IL:|fetch| BELOW? IL:|of| CONTEXT))
                             (IL:|replace| CURRENT-BLOCK IL:|of| CONTEXT IL:|with| BLOCK))
            IL:|do| (COND
                       (BELOW? (WHEN (EQ BELOW? 'NEW)
                                     (IL:|replace| LINEAR-POINTER IL:|of| CONTEXT IL:|with| TEMP)
                                     (REPAINT-NEW-LINE (IL:|fetch| PREV-LINE IL:|of| (CAR TEMP))))
                               (IL:SETQ TEMP (CAR TEMP)))
                       (T (IL:|replace| BLOCK-START IL:|of| BLOCK IL:|with| (CDR TEMP))
                          (IL:|replace| CURRENT-LINE IL:|of| CONTEXT IL:|with| TEMP)
                          (IL:|replace| SHIFT-Y IL:|of| CONTEXT IL:|with| NIL)
                          (IL:SETQ TEMP (CAR TEMP))
                          (IL:|replace| BLOCK-X IL:|of| BLOCK IL:|with| (IL:IDIFFERENCE (IL:|fetch| INDENT IL:|of| TEMP)
                                                                                        DELTA-X))
                          (IL:|replace| BLOCK-NEW-X IL:|of| BLOCK IL:|with| (IL:|fetch| INDENT IL:|of| TEMP))
                          (IL:|replace| BLOCK-ASCENT IL:|of| BLOCK IL:|with| (IL:|fetch| LINE-ASCENT IL:|of| TEMP))
                          (IL:|replace| BLOCK-DESCENT IL:|of| BLOCK IL:|with| (IL:|fetch| LINE-DESCENT IL:|of| TEMP))
                          (IL:|replace| BLOCK-BASE-LINE IL:|of| BLOCK IL:|with| (IL:|fetch| BASE-LINE-Y IL:|of| TEMP))
                          (IL:|replace| BITS? IL:|of| BLOCK IL:|with| T)))
                    (IL:|replace| YCOORD IL:|of| TEMP IL:|with| (IL:IPLUS (IL:|fetch| YCOORD IL:|of| TEMP)
                                                                          DELTA-Y))
                    (WHEN (EQ TEMP (IL:|fetch| LAST-LINE IL:|of| NODE))
                        (WHEN (NOT BELOW?)
                            (IL:|replace| CACHED-Y IL:|of| TEMP IL:|with| (IL:IDIFFERENCE (IL:|fetch| BASE-LINE-Y
                                                                                               IL:|of| TEMP)
                                                                                          DELTA-Y))
                            (IL:|replace| CACHED-ASCENT IL:|of| TEMP IL:|with| (IL:|fetch| LINE-ASCENT IL:|of| TEMP))
                            (IL:|replace| CACHED-DESCENT IL:|of| TEMP IL:|with| (IL:|fetch| LINE-DESCENT IL:|of| TEMP))
                            (IL:|replace| CACHE-TIME IL:|of| TEMP IL:|with| (IL:|fetch| RELINEARIZATION-TIME-STAMP
                                                                                IL:|of| CONTEXT)))
                        (RETURN))
                    (IL:|replace| LINE-LENGTH IL:|of| TEMP IL:|with| (IL:SETQ CURRENT-X (IL:IPLUS (IL:|fetch|
                                                                                                    LINE-LENGTH
                                                                                                  IL:|of| TEMP)
                                                                                             DELTA-X)))
                    (IL:SETQ TEMP (IL:|fetch| NEXT-LINE IL:|of| TEMP))
                    (IL:|add| (IL:|fetch| INDENT IL:|of| (CAR TEMP))
                           DELTA-X)
                    (WHEN (NOT BELOW?)
                        (LINE-FINISHED CONTEXT CURRENT-X TEMP)
                        (IL:SETQ BELOW? (IL:|fetch| BELOW? IL:|of| CONTEXT))))
                                                            ; used to be replace CurrentLine of context with (SETQ
                                                            ; current.line (fetch LastLineLinear of node))
         (IL:|replace| CURRENT-LINE IL:|of| CONTEXT IL:|with| (IL:SETQ CURRENT-LINE (LAST-LINE-LINEAR NODE CONTEXT)
                                                              ))
         (IL:SETQ CURRENT-X (IL:IPLUS (IL:|fetch| INDENT IL:|of| TEMP)
                                      (SCAN-FOR-BOUNDS (CDR CURRENT-LINE)
                                             (CDR (LAST (IL:|fetch| LINEAR-FORM IL:|of| NODE)))
                                             CURRENT-LINE T)))))
(WHEN (IL:NEQ CURRENT-X (IL:IPLUS (IL:|fetch| START-X IL:|of| NODE)
                                  (IL:|fetch| ACTUAL-LLENGTH IL:|of| NODE)))
      (IL:SHOULDNT "old ActualLLength value doesn't match"))
(IL:|replace| CURRENT-X IL:|of| CONTEXT IL:|with| CURRENT-X)
(IL:|replace| LINEAR-POINTER IL:|of| CONTEXT IL:|with| (CDR (IL:|replace| LINEAR-PREV IL:|of| CONTEXT
                                                                IL:|with| (IL:|fetch| LINEAR-THREAD IL:|of| NODE)))))))
```

(SHIFT-BLOCK

```
(IL:LAMBDA (CONTEXT X Y WIDTH START END ASCENT DESCENT NEW-X OLD-Y)
                                                                    ; Edited 17-Nov-87 11:45 by DCB

;;; we've found a block of bits in the window which can be reused.  bitblt them to the appropriate place

    (LET* ((CURRENT-LINE (CAR (IL:FETCH CURRENT-LINE IL:OF CONTEXT)))
           (CURRENT-LINE-BOTTOM (IL:ADD1 (IL:FETCH NEXT-LINE-Y IL:OF CURRENT-LINE)))
           (DELTA (IL:IDIFFERENCE (IL:IDIFFERENCE Y DESCENT)
                                  CURRENT-LINE-BOTTOM))
           (REPAINT-START (IL:FETCH REPAINT-START IL:OF CONTEXT))
           H W)
          (WHEN (AND (IL:IGREATERP DELTA 0)
                     (IL:IGREATERP (IL:SETQ H (IL:IDIFFERENCE CURRENT-LINE-BOTTOM (IL:FETCH WINDOW-BOTTOM
                                                                                    IL:OF CONTEXT)))
                                   0))
              ;; we're shifting stuff down, so move the bits below them out of the way (down) first
              (IL:BITBLT (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                     (IL:FETCH WINDOW-LEFT IL:OF CONTEXT)
                     (IL:IPLUS (IL:FETCH WINDOW-BOTTOM IL:OF CONTEXT)
                            DELTA)
                     (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                     (IL:FETCH WINDOW-LEFT IL:OF CONTEXT)
                     (IL:FETCH WINDOW-BOTTOM IL:OF CONTEXT)
                     (IL:ADD1 (IL:IDIFFERENCE (IL:FETCH WINDOW-RIGHT IL:OF CONTEXT)
                                   (IL:FETCH WINDOW-LEFT IL:OF CONTEXT)))
                     H)
              (IL:REPLACE SHIFT-DOWN IL:OF CONTEXT IL:WITH (IL:IPLUS (IL:FETCH SHIFT-DOWN IL:OF CONTEXT)
                                                                     DELTA)))
          (WHEN (OR (IL:NEQ Y (IL:FETCH BASE-LINE-Y IL:OF CURRENT-LINE))
                    (IL:NEQ X NEW-X))
              ;; the bits aren't already in the right place, so move them
              (COND
                 ((IL:IGREATERP (IL:IPLUS Y ASCENT)
                         CURRENT-LINE-BOTTOM)
                  ;; we'll take along the rest of the line while we're at it (rather than lose those bits)
                  (COND
                     ((EQ OLD-Y (IL:FETCH SHIFT-Y IL:OF CONTEXT))
                      (IL:REPLACE SHIFT-RIGHT IL:OF CONTEXT IL:WITH (IL:IPLUS (IL:FETCH SHIFT-RIGHT IL:OF CONTEXT)
                                                                              (IL:IDIFFERENCE NEW-X X))))
                     (T (IL:REPLACE SHIFT-RIGHT IL:OF CONTEXT IL:WITH (IL:IDIFFERENCE NEW-X X))
                        (IL:REPLACE SHIFT-Y IL:OF CONTEXT IL:WITH OLD-Y)))
                  (IL:SETQ W (IL:ADD1 (IL:IDIFFERENCE (IL:FETCH WINDOW-RIGHT IL:OF CONTEXT)
                                                      NEW-X))))
                 (T (IL:SETQ W WIDTH)))
              (IL:SETQ DESCENT (IL:IMIN DESCENT (IL:FETCH LINE-DESCENT IL:OF CURRENT-LINE)))
              (IL:SETQ ASCENT (IL:IMIN ASCENT (IL:FETCH LINE-ASCENT IL:OF CURRENT-LINE)))
              (IL:BITBLT (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                     X
                     (IL:IDIFFERENCE Y DESCENT)
                     (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                     NEW-X
                     (IL:IDIFFERENCE (IL:FETCH BASE-LINE-Y IL:OF CURRENT-LINE)
                            DESCENT)
                     W
                     (IL:IPLUS ASCENT DESCENT)))
          (WHEN (IL:ILEQ (IL:SETQ ASCENT (IL:IPLUS (IL:FETCH BASE-LINE-Y IL:OF CURRENT-LINE)
                                                   ASCENT))
                         (IL:FETCH YCOORD IL:OF CURRENT-LINE))
              ;; it wasn't as tall as the line it's moved to, so blank above it
              (IL:BLTSHADE IL:WHITESHADE (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                     NEW-X ASCENT WIDTH (IL:ADD1 (IL:IDIFFERENCE (IL:FETCH YCOORD IL:OF CURRENT-LINE)
                                                                 ASCENT))))
          (WHEN (IL:ILESSP DESCENT (IL:FETCH LINE-DESCENT IL:OF CURRENT-LINE))
              ;; it descend as mush as the line it's moved to, so blank below it
              (IL:BLTSHADE IL:WHITESHADE (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                     NEW-X
                     (IL:ADD1 (IL:FETCH NEXT-LINE-Y IL:OF CURRENT-LINE))
                     WIDTH
                     (IL:IDIFFERENCE (IL:FETCH LINE-DESCENT IL:OF CURRENT-LINE)
                            DESCENT)))
          (WHEN (IL:TYPE? LINE-START (CAR REPAINT-START))
              (IL:SETQ REPAINT-START (CDR REPAINT-START)))
          (COND
             ((EQ REPAINT-START START)
              ;; nothing to be painted, just blank where necessary
              (WHEN (AND (OR (IL:NEQ Y (IL:FETCH BASE-LINE-Y IL:OF CURRENT-LINE))
                             (IL:NEQ X NEW-X))
                         (EQ NEW-X (IL:FETCH INDENT IL:OF CURRENT-LINE)))
```

```
                    ;; this is the start of the line, so blank to the left margin
                    (IL:BLTSHADE IL:WHITESHADE (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                           (IL:FETCH WINDOW-LEFT IL:OF CONTEXT)
                           (IL:ADD1 (IL:FETCH NEXT-LINE-Y IL:OF CURRENT-LINE))
                           (IL:IDIFFERENCE NEW-X (IL:FETCH WINDOW-LEFT IL:OF CONTEXT))
                           (IL:FETCH LINE-HEIGHT IL:OF CURRENT-LINE))))
               (T ;; there is extra material to paint in front of the bits we've moved
                  (COND
                     ((IL:NEQ (IL:FETCH REPAINT-LINE IL:OF CONTEXT)
                              CURRENT-LINE)
                      ;; there are several lines of stuff to paint.  blank the area it's going to first
                      (IL:BLTSHADE IL:WHITESHADE (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                             (IL:FETCH WINDOW-LEFT IL:OF CONTEXT)
                             (IL:ADD1 (IL:FETCH YCOORD IL:OF CURRENT-LINE))
                             (IL:ADD1 (IL:IDIFFERENCE (IL:FETCH WINDOW-RIGHT IL:OF CONTEXT)
                                                      (IL:FETCH WINDOW-LEFT IL:OF CONTEXT)))
                             (IL:IDIFFERENCE (IL:FETCH YCOORD IL:OF (IL:FETCH REPAINT-LINE IL:OF CONTEXT))
                                             (IL:FETCH YCOORD IL:OF CURRENT-LINE)))
                      (IL:BLTSHADE IL:WHITESHADE (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                             (IL:FETCH WINDOW-LEFT IL:OF CONTEXT)
                             (IL:ADD1 (IL:FETCH NEXT-LINE-Y IL:OF CURRENT-LINE))
                             (IL:IDIFFERENCE NEW-X (IL:FETCH WINDOW-LEFT IL:OF CONTEXT))
                             (IL:FETCH LINE-HEIGHT IL:OF CURRENT-LINE))
                      (REPAINT CONTEXT (IL:FETCH REPAINT-X IL:OF CONTEXT)
                             (IL:FETCH BASE-LINE-Y IL:OF (IL:FETCH REPAINT-LINE IL:OF CONTEXT))
                             REPAINT-START START))
                     (T ;; the stuff to be repainted is all on this line
                        (WHEN (EQ (IL:SETQ X (IL:FETCH REPAINT-X IL:OF CONTEXT))
                                  (IL:FETCH INDENT IL:OF (IL:FETCH REPAINT-LINE IL:OF CONTEXT)))
                            ;; this is the beginning of the line, so blank to the left margin
                            (IL:SETQ X (IL:FETCH WINDOW-LEFT IL:OF CONTEXT)))
                        (IL:BLTSHADE IL:WHITESHADE (IL:FETCH DISPLAY-WINDOW IL:OF CONTEXT)
                               X
                               (IL:ADD1 (IL:FETCH NEXT-LINE-Y IL:OF CURRENT-LINE))
                               (IL:IDIFFERENCE NEW-X X)
                               (IL:FETCH LINE-HEIGHT IL:OF CURRENT-LINE))
                        (REPAINT CONTEXT (IL:FETCH REPAINT-X IL:OF CONTEXT)
                               (IL:FETCH BASE-LINE-Y IL:OF (IL:FETCH REPAINT-LINE IL:OF CONTEXT))
                               REPAINT-START START)))))
            (IL:REPLACE REPAINT-START IL:OF CONTEXT IL:WITH END)
            (IL:REPLACE REPAINT-LINE IL:OF CONTEXT IL:WITH CURRENT-LINE)
            (IL:REPLACE REPAINT-X IL:OF CONTEXT IL:WITH (IL:IPLUS NEW-X WIDTH)))))
```

(TRY-REUSING-BITS

```
  (IL:LAMBDA (CONTEXT BLOCK)                                    ; Edited 17-Nov-87 11:47 by DCB
```

;;; decide whether the bits described by this block are actually available. if so, use shift.block to move them to the appropriate position. return T if we
;;; were successful

```
    (PROG ((SHIFTED-X (IL:FETCH BLOCK-X IL:OF BLOCK))
           (LEFT-CLIP (IL:FETCH WINDOW-LEFT IL:OF CONTEXT))
           (START (IL:FETCH BLOCK-START IL:OF BLOCK))
           (END (IL:FETCH BLOCK-START IL:OF (IL:FETCH NEXT-BLOCK IL:OF BLOCK)))
           (ASCENT (IL:FETCH BLOCK-ASCENT IL:OF BLOCK))
           (DESCENT (IL:FETCH BLOCK-DESCENT IL:OF BLOCK))
           (WIDTH (IL:FETCH BLOCK-WIDTH IL:OF BLOCK))
           (NEW-X (IL:FETCH BLOCK-NEW-X IL:OF BLOCK))
           SHIFTED-Y)
          (WHEN (EQ WIDTH 0)
              ;; no point in it if there are no bits
              (GO NO-GOOD))
          ;; make sure they haven't been overwritten already, or shifted off the window
          (COND
             ((IL:FETCH SHIFT-Y IL:OF CONTEXT)
              (COND
                 ((EQ (IL:FETCH BLOCK-BASE-LINE IL:OF BLOCK)
                      (IL:FETCH SHIFT-Y IL:OF CONTEXT))
                  (IL:SETQ SHIFTED-Y (IL:FETCH BASE-LINE-Y IL:OF (CAR (IL:FETCH CURRENT-LINE IL:OF CONTEXT))))
                  (IL:SETQ SHIFTED-X (IL:IPLUS SHIFTED-X (IL:FETCH SHIFT-RIGHT IL:OF CONTEXT)))
                  (IL:SETQ LEFT-CLIP (IL:FETCH REPAINT-X IL:OF CONTEXT)))
                 (T (IL:SETQ SHIFTED-Y (IL:IDIFFERENCE (IL:FETCH BLOCK-BASE-LINE IL:OF BLOCK)
                                                       (IL:FETCH SHIFT-DOWN IL:OF CONTEXT)))
                    (WHEN (IL:IGREATERP (IL:SUB1 (IL:IPLUS SHIFTED-Y ASCENT))
                                        (IL:FETCH NEXT-LINE-Y IL:OF (CAR (IL:FETCH CURRENT-LINE IL:OF CONTEXT))))
                          (GO NO-GOOD)))))
             (T (IL:SETQ SHIFTED-Y (IL:IDIFFERENCE (IL:FETCH BLOCK-BASE-LINE IL:OF BLOCK)
                                                   (IL:FETCH SHIFT-DOWN IL:OF CONTEXT)))
                (WHEN (IL:IGREATERP (IL:SUB1 (IL:IPLUS SHIFTED-Y ASCENT))
                                    (IL:FETCH YCOORD IL:OF (IL:FETCH REPAINT-LINE IL:OF CONTEXT)))
```

```
                    (GO NO-GOOD))
                (WHEN (IL:IGREATERP (IL:SUB1 (IL:IPLUS SHIFTED-Y ASCENT))
                            (IL:FETCH NEXT-LINE-Y IL:OF (IL:FETCH REPAINT-LINE IL:OF CONTEXT)))
                    (IL:SETQ LEFT-CLIP (IL:FETCH REPAINT-X IL:OF CONTEXT)))))
            (WHEN (OR (IL:IGREATERP SHIFTED-X (IL:FETCH WINDOW-RIGHT IL:OF CONTEXT))
                      (IL:ILEQ (IL:IPLUS SHIFTED-Y ASCENT)
                               (IL:FETCH WINDOW-BOTTOM IL:OF CONTEXT)))
                                                                    ; none of it's within the window
                (GO NO-GOOD))
            (WHEN (OR (AND (IL:IGREATERP (IL:SUB1 (IL:IPLUS SHIFTED-Y ASCENT))
                                   (IL:FETCH WINDOW-TOP IL:OF CONTEXT))
                           (IL:ILESSP (IL:FETCH BASE-LINE-Y IL:OF (CAR (IL:FETCH CURRENT-LINE IL:OF CONTEXT)))
                                   SHIFTED-Y))
                      (AND (IL:ILESSP (IL:IDIFFERENCE SHIFTED-Y DESCENT)
                                   (IL:FETCH WINDOW-BOTTOM IL:OF CONTEXT))
                           (IL:IGREATERP (IL:FETCH BASE-LINE-Y IL:OF (CAR (IL:FETCH CURRENT-LINE IL:OF CONTEXT)))
                                   SHIFTED-Y)))
                ;; some of it's within the window, but too much is clipped by the top or bottom edge of the window
                (GO NO-GOOD))
            (WHEN (IL:IGEQ LEFT-CLIP (IL:IPLUS SHIFTED-X WIDTH))
                  (GO NO-GOOD))
            (IL:SETQ START (NEXT-LINEAR-ITEM START))
            (WHEN (IL:IGREATERP (IL:IDIFFERENCE LEFT-CLIP SHIFTED-X)
                          (IL:IDIFFERENCE (IL:FETCH WINDOW-RIGHT IL:OF CONTEXT)
                                 NEW-X))
                ;; this block was clipped on the left, so adjust its description (we'll have to repaint more)
                (IL:WHILE (IL:IGREATERP LEFT-CLIP SHIFTED-X) IL:BIND W IL:DO (IL:SETQ W (LINEAR-ITEM-WIDTH
                                                                                         (CAR START)))
                                                                             (WHEN (IL:IGEQ W WIDTH)
                                                                                                    ; there's nothing useable left
                                                                                 (GO NO-GOOD))
                                                                             (IL:SETQ WIDTH (IL:IDIFFERENCE WIDTH W)
                                                                              )
                                                                             (IL:SETQ SHIFTED-X (IL:IPLUS SHIFTED-X
                                                                                                          W))
                                                                             (IL:SETQ NEW-X (IL:IPLUS NEW-X W))
                                                                             (IL:SETQ START (NEXT-LINEAR-ITEM
                                                                                             (CDR START)))))
            (WHEN (AND (IL:IGREATERP SHIFTED-X NEW-X)
                       (IL:IGREATERP (IL:SUB1 (IL:IPLUS SHIFTED-X WIDTH))
                              (IL:FETCH WINDOW-RIGHT IL:OF CONTEXT)))
                ;; this block was clipped on the right, so adjust its description (we'll have to repaint more)
                (IL:SETQ END START)
                (IL:SETQ WIDTH 0)
                (IL:BIND W IL:UNTIL (IL:IGREATERP (IL:SUB1 (IL:IPLUS SHIFTED-X WIDTH (IL:SETQ W (LINEAR-ITEM-WIDTH
                                                                                                  (CAR END)))))
                                                  (IL:FETCH WINDOW-RIGHT IL:OF CONTEXT))
                   IL:DO (IL:SETQ WIDTH (IL:IPLUS WIDTH W))
                         (IL:SETQ END (NEXT-LINEAR-ITEM (CDR END))))
                (WHEN (EQ START END)
                    ;; there's nothing useable left
                    (GO NO-GOOD)))
        ;; there seem to be some useful bits here.  put them in the right place
        (SHIFT-BLOCK CONTEXT SHIFTED-X SHIFTED-Y WIDTH START END ASCENT DESCENT NEW-X (IL:FETCH
                                                                                          BLOCK-BASE-LINE
                                                                                   IL:OF BLOCK))
        (RETURN T)
     NO-GOOD)))

)

(IL:PUTPROPS IL:SEDIT-LINEAR IL:COPYRIGHT ("Venue & Xerox Corporation" 1986 1987 1988 1990))
```

FUNCTION INDEX

PROPERTY INDEX