

File created: 6-Jan-92 15:22:46 {DSK}<usr>local>lde>lispcore>sources>SEdit-TOPLEVEL.;2

changes to: (IL:FNS COMPLETION)
(IL:VARS IL:SEdit-TOPLEVELCOMS)
(IL:FUNCTIONS FIX-EDITDATE)

previous date: 10-Jul-91 19:11:12 {DSK}<usr>local>lde>lispcore>sources>SEdit-TOPLEVEL.;1

Read Table: XCL

Package: SEDIT

Format: XCCS

; Copyright (c) 1986, 1987, 1988, 1990, 1991, 1992 by Venue & Xerox Corporation. All rights reserved.

```
(IL:RPAQQ IL:SEdit-TOPLEVELCOMS
  ((IL:PROP IL:FILETYPE IL:SEdit-TOPLEVEL)
   (IL:PROP IL:MAKEFILE-ENVIRONMENT IL:SEdit-TOPLEVEL)
   (IL:LOCALVARS . T)
   (IL:DECLARE\ : IL:DONTCOPY IL:DOEVAL@COMPILE (IL:FILES IL:SEdit-DECLS))
   (IL:INITVARS CONTEXTS REGIONS)
   (IL:VARS (IL:*DISPLAY-EDITOR* 'SEdit))
   (IL:FNS SEDIT RESET GET-WINDOW-REGION SAVE-WINDOW-REGION)
   (IL:FNS GET-CONTEXT DISINTEGRATE-CONTEXT AWAKE-COMMAND-PROCESS AWAKE-ME MARKASCHANGEDFN
    NEW-FUNCTION-BODY)
   (IL:FUNCTIONS QUERY-THROW-AWAY-CHANGES SET-OPTIONS SET-PROPS START-PROCESS)
   (IL:COMS

    ;; THESE CAN ALL BE NUKED WITH THE NEW EDIT INTERFACE AND A DETACHED TTY/EDITOR (WOZ 1/25/91)
    (IL:PROP (IL:|Definition-for-EDITL| IL:|Definition-for-EDITE| IL:|Definition-for-EDITDATE|)
     SEDIT)
    (IL:FNS SEDIT SEDITL FN-CHANGED PROP-CHANGED PROPLST-CHANGED VAR-CHANGED ALIST-COMPLETION
     COMPLETION PROPS-COMPLETION TTYFN LOCATE-NODE-FROM-EDITCHAIN)

    ;; SEdit's hack way of fixing edit dates
    (IL:FUNCTIONS FIX-EDITDATE)

    ;; Mess around with the tty editor's TTY: command by defining a hook and then making TTY: a macro which calls the hook.
    (IL:FUNCTIONS SMART-TTYFN)
    (IL:P (PUSHNEW ' (IL:TTY\ : NIL (IL:E (SMART-TTYFN)
      T)))
      IL:EDITMACROS :TEST #'IL:EQUAL)))
  (IL:FNS PRETTY-PRINT MAP-FONT)
  ;; these guys allow you to print and read structures with broken atoms and gaps. just a convenience for the loser who forgets to get them
  ;; out of his code.
  (IL:FUNCTIONS MAKE-BROKEN-ATOM PRINT-BROKEN-ATOM MAKE-GAP PRINT-GAP)
  (IL:P (IL:DEFPRINT 'BROKEN-ATOM 'PRINT-BROKEN-ATOM)
   (IL:DEFPRINT 'GAP 'PRINT-GAP))))

(IL:PUTPROPS IL:SEdit-TOPLEVEL IL:FILETYPE :COMPILE-FILE)

(IL:PUTPROPS IL:SEdit-TOPLEVEL IL:MAKEFILE-ENVIRONMENT (:READTABLE "XCL" :PACKAGE (DEFPACKAGE "SEdit"
  (:USE "LISP" "XCL"))))

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

(IL:LOCALVARS . T)
)

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

(IL:FILESLOAD IL:SEdit-DECLS)
)

(IL:RPAQ? CONTEXTS NIL)

(IL:RPAQ? REGIONS NIL)

(IL:RPAQQ IL:*DISPLAY-EDITOR* SEDIT)

(IL:DEFINEQ

(SEdit
  (IL:LAMBDA (STRUCTURE PROPS OPTIONS)
    (OR STRUCTURE (IL:SETQ STRUCTURE BASIC-GAP))
    (LET* ((NAME (IL:LISTGET PROPS :NAME))
      (TYPE (OR (IL:LISTGET PROPS :TYPE)
        :EXPRESSION))
      (CONTEXT (GET-CONTEXT STRUCTURE NAME TYPE))
      (WINDOW (IL:|fetch| DISPLAY-WINDOW IL:|of| CONTEXT)))
      (SET-PROPS CONTEXT PROPS)
      (SET-OPTIONS CONTEXT OPTIONS)
      (COND
        ((NULL WINDOW)
         ;; this is a new context, needs to be setup from scratch
```

; Edited 25-Jan-91 13:51 by woz

```

    (START-PROCESS CONTEXT NAME)
    CONTEXT)
  (AND (IL:OPENWP WINDOW)
        (IL:PROCESSP (IL:WINDOWPROP WINDOW 'IL:PROCESS)))
  ;; open and active
  (IL:TOTOPW WINDOW)
  (WHEN (OR (NOT (EQ T (IL:|fetch| (EDIT-CONTEXT CHANGED-STRUCTURE?) IL:|of| CONTEXT)))
            (QUERY-THROW-AWAY-CHANGES NAME OPTIONS))
    ;; there are no changes on this edit, or user said throw away changes, so we will restart this edit.
    (IL:|replace| CHANGED-STRUCTURE? IL:|of| CONTEXT IL:|with| STRUCTURE)
    (IL:RESTART.PROCESS (IL:WINDOWPROP WINDOW 'IL:PROCESS))
    CONTEXT))
  (IL:OPENWP (IL:WINDOWPROP WINDOW 'IL:ICONWINDOW))
  ;; shrunk
  (IL:|replace| CHANGED-STRUCTURE? IL:|of| CONTEXT IL:|with| STRUCTURE)
  (IL:EXPANDW WINDOW)
  CONTEXT)
  (T ;; found a dead context. get rid of it and try again.
    (DISINTEGRATE-CONTEXT CONTEXT)
    (SEdit STRUCTURE PROPS OPTIONS))))))

```

(RESET

```

  (IL:LAMBDA NIL ; Edited 10-Jul-87 08:35 by DCB
    (COND
      (CONTEXTS (IL:ERROR "Can't reset SEdit while there are open SEdit windows"))
      (T (CREATE-ENVIRONMENTS)
          (RESET-FORMATS)
          T))))

```

(GET-WINDOW-REGION

```

  (IL:LAMBDA (CONTEXT REASON NAME TYPE) ; Edited 19-Nov-87 10:18 by DCB

```

;; called to get a region for this sedit window. should return the region for the sedit including the prompt window. context is being built and needs a window. the context will have at least the name (IconTitle) and type (EditType) of the object being edited, and can be used as desired to map between contexts and windows. If reason is :CREATE, then this function must return a region. If :EXPAND, then this algorithm returns a region from the stack only if SEDIT.KEEP.WINDOW.REGION is nil, otherwise it returns NIL, telling the window system not to reshape on expansion.

```

  (WHEN (OR (EQ REASON :CREATE)
            (NOT KEEP-WINDOW-REGION))
    (OR (IL:POP REGIONS)
        (PROGN (IL:|printout| IL:PROMPTWINDOW T "Select region for SEdit window.")
                (IL:GETREGION 30 20)))))

```

(SAVE-WINDOW-REGION

```

  (IL:LAMBDA (CONTEXT REASON NAME TYPE REGION) ; Edited 23-Nov-87 17:46 by DCB

```

;; Release this sedit windows region to be used again. If we're shrinking, KEEP-WINDOW-REGION determines whether to release the region or not. If an icon is being closed, don't release the region because it was handled appropriately when the window as shrunk. remember, we're maintaining regions including the prompt window height, so use WINDOWREGION to get the whole region.

```

  (WHEN (OR (EQ REASON :CLOSE)
            (AND (EQ REASON :SHRINK)
                 (NOT KEEP-WINDOW-REGION)))
    (IL:|push| REGIONS (OR REGION (IL:WINDOWREGION (IL:|fetch| DISPLAY-WINDOW IL:|of| CONTEXT))))))

```

```

)

```

```

(IL:DEFINEQ

```

(GET-CONTEXT

```

  (IL:LAMBDA (STRUCTURE NAME TYPE SEARCH-ONLY?) ; Edited 5-Dec-90 13:00 by woz

```

;; we've been asked to get the edit context for a new edit. if a context matching this description (same name and same type, or EQ structure) already exists, we'll return it rather than creating a new one. Also, if SEARCH-ONLY? is true then don't create a new one, just return NIL if not found.

```

  (IL:|bind| WINDOW IL:|for| CONTEXT IL:|in| CONTEXTS
    IL:|when| (OR (AND NAME (EQUAL NAME (IL:|fetch| ICON-TITLE IL:|of| CONTEXT))
                     (EQ TYPE (IL:|fetch| EDIT-TYPE IL:|of| CONTEXT)))
                (AND STRUCTURE (IL:|type?| EDIT-NODE (IL:|fetch| ROOT IL:|of| CONTEXT))
                     (EQ STRUCTURE (IL:|fetch| STRUCTURE IL:|of| (SUBNODE 1 (IL:|fetch| ROOT IL:|of| CONTEXT))))))
    IL:|do|

```

```

      ;; we found a context that matches, return it.

```

```

      (RETURN CONTEXT)

```

```

  IL:|finally|

```

```

    ;; this is a new editing task, so make an appropriate context and get it started

```

```

    (IF SEARCH-ONLY?
      (RETURN NIL)
      (LET ((CONTEXT (IL:|create| EDIT-CONTEXT

```

```

COMPLETION-EVENT IL:_ (IL:CREATE.EVENT (IL:CONCAT EDITOR-NAME NAME))
ROOT IL:_ STRUCTURE
ICON-TITLE IL:_ NAME
EDIT-TYPE IL:_ TYPE)))
(PUSH CONTEXT CONTEXTS)
(RETURN CONTEXT)))))

```

(DISINTEGRATE-CONTEXT

(IL:LAMBDA (CONTEXT)

; Edited 5-Dec-90 17:45 by woz

```

;;; terminate this edit context. we mark it as dead, remove it from the active edits list, smash the connections between the context and the window

```

```

(WHEN CONTEXT
  (IL:NOTIFY.EVENT (IL:|fetch| COMPLETION-EVENT IL:|of| CONTEXT))
  (IL:|replace| CONTEXT-LOCK IL:|of| CONTEXT IL:|with| 'DEAD)
  (IL:WINDOWPROP (IL:|fetch| DISPLAY-WINDOW IL:|of| CONTEXT)
    'EDIT-CONTEXT NIL)
  (IL:|replace| DISPLAY-WINDOW IL:|of| CONTEXT IL:|with| NIL)
  (IL:SETQ CONTEXTS (IL:DREMOVE CONTEXT CONTEXTS)))))

```

(AWAKE-COMMAND-PROCESS

(IL:LAMBDA (CONTEXT COMMAND)

; Edited 5-Dec-90 16:52 by woz

```

;; if this context has a process associated with it, and the process is currently stuck waiting for input, unstick it so that it can look around and
;; (presumably) notice some important change in its environment. This is also called when someone in another process, such as a command menu
;; or a window menu operation, wants to tell the command process to execute the command. Note that under a few circumstances this function will
;; be called by a running command in the sedit process. For example, the complete-and-close command calls il:closew which calls sedit's closefn
;; which tries to wake up the sedit to let it know the window was closed. In this case, awake-command-process will result in a no-op because sedit
;; has a command running, and therefore cannot be woken up. COMMAND is a command form which will be used as the value returned from
;; GETKEY in SEDIT1. COMMAND should be of the form (<fn> <normalize?> <extra args>), so that <fn> will be applied to the context, the
;; charcode invokeing the command (NIL in this case), and the extra args. After the command runs the window will scroll to normalize the caret if
;; <normalize?> is T. If COMMAND is NIL then the SEdit will just update the window.

```

```

(LET ((PROCESS (IL:WINDOWPROP (IL:|fetch| DISPLAY-WINDOW IL:|of| CONTEXT)
  'IL:PROCESS)))
  (WHEN (IL:PROCESSP PROCESS)
    (IL:PROCESS.APPLY PROCESS 'AWAKE-ME (LIST COMMAND)))))

```

(AWAKE-ME

(IL:LAMBDA (RESULT)

; Edited 7-Jul-87 12:59 by DCB

```

;; this rather ugly little function checks to see if it's being called under getkey (presumably by PROCESS.APPLY from awake.command.process)
;; and if so forces the getkey to return result

```

```

(LET ((STACK-FRAME (IL:STKPOS 'GETKEY)))
  (WHEN STACK-FRAME (IL:RETFROM STACK-FRAME RESULT T)))))

```

(MARKASCHANGEDFN

(IL:LAMBDA (NAME TYPE REASON)

; Edited 3-Apr-91 15:42 by jds

```

;;; When a managed object is changed, we must check if we have an open edit on it. If so, calling SEdit again, with the fresh definition, will force the
;;; update. This is fairly tricky, though. Markaschanged is called as a result of editing a managed definition, so this markaschangedfn could be running
;;; in the sedit process under the completion-fn half way through completion. IDEALLY in this case we could say "i know it changed, i just changed it!"
;;; and ignore this call. BUT FOR NOW (1/14/91) since the manager can change the definition on completion (editdates, for one), we have to notify
;;; SEdit. Since calling editdef will restart the sedit process, the completion-fn will not finish, so do the verify-structure here.

```

```

(LET* ((FORM (IL:PROCESSPROP (IL:THIS.PROCESS)
  'IL:FORM))
  CONTEXT)
  (COND
    ((AND (EQ (CAR FORM)
      'SEDIT1)
      (IL:|type?| EDIT-CONTEXT (SETQ CONTEXT (CADADR FORM)))
      (EQ NAME (IL:|fetch| ICON-TITLE IL:|of| CONTEXT))
      (EQ TYPE (IL:|fetch| EDIT-TYPE IL:|of| CONTEXT)))
      ;; we're running under the edit that is completing
      (UNLESS *IGNORE-CHANGES-ON-COMPLETION*
        (VERIFY-STRUCTURE CONTEXT NIL (IL:GETDEF NAME TYPE NIL ' (IL:EDIT IL:NOCOPY)))))
      ((GET-CONTEXT NIL NAME TYPE T)
      ;; found a matching context elsewhere
      (IL:EDITDEF NAME TYPE NIL NIL ' (:DONTWAIT))))))

```

(NEW-FUNCTION-BODY

(IL:LAMBDA (DUMMY-BODY)

; Edited 7-Jul-87 12:59 by DCB

```

  (IF (IL:NEQ (IL:EDITMODE)
    'SEDIT)
    (IL:COPY DUMMY-BODY)
    (LIST 'IL:LAMBDA ARGS-GAP BODY-GAP))))

```

)

```
(DEFUN QUERY-THROW-AWAY-CHANGES (NAME OPTIONS)
```

```
;; this gets called when sedit is restarting because it got called again, but the structure doesn't match and changes have been made. should we throw
;; away the changes and restart with the new structure, or not restart and keep the changes? ask the user.
```

```
(IF (IL:EQMEMB :DISPLAY OPTIONS)
  (IL:MENUS (IL:|create| IL:MENUS
    (IL:ITEMS IL:_ ' ("Throw away changes and restart with new structure" T)
      ("Keep changes and don't restart with new structure" NIL))
    (IL:TITLE IL:_ (FORMAT NIL "An edit session with changes already exists for ~S" NAME)))
  (IF (EQ 'IL:Y (IL:ASKUSER NIL NIL (FORMAT NIL "An edit session with changes already exists for ~S. Throw
    away changes and restart with new structure? " NAME)))
    T)))
```

```
(DEFUN SET-OPTIONS (CONTEXT OPTIONS)
```

```
;; set up the OPTIONS provided in the call to SEDIT for this context. Most of these options do not require immediate action. Rather, they control how
;; some command or interaction will work later, so we just store the option list in the context. Most of these options are really edit-interface options, not
;; sedit options. We stash them so that when the *edit-fn* is called under M-O, it will be handed the same options that this edit was started with
```

```
(IL:REPLACE (EDIT-CONTEXT EDIT-OPTIONS) IL:OF CONTEXT IL:WITH (IF (LISTP OPTIONS)
  OPTIONS
  (LIST OPTIONS)))
```

```
(DEFUN SET-PROPS (CONTEXT PROPS)
```

```
;; go through the PROPS list supplied in the call to SEDIT and store the info in the context. The :NAME and :TYPE props are already handled, because
;; get-context uses this information to find an appropriate context. Grab the current values of the variables that determine reading and printing, and save
;; them in a profile in the context, so that later changes to the globals don't affect existing contexts.
```

```
(IL:REPLACE (EDIT-CONTEXT COMPLETION-FN) IL:OF CONTEXT IL:WITH (OR (IL:LISTGET PROPS :COMPLETION-FN)
  #'NULL))
(IL:REPLACE (EDIT-CONTEXT ROOT-CHANGED-FN) IL:OF CONTEXT IL:WITH (OR (IL:LISTGET PROPS :ROOT-CHANGED-FN)
  #'NULL))
(IL:REPLACE (EDIT-CONTEXT ENVIRONMENT) IL:OF CONTEXT IL:WITH (OR (IL:LISTGET PROPS :ENVIRONMENT)
  LISP-EDIT-ENVIRONMENT))
(IL:REPLACE (EDIT-CONTEXT PROFILE) IL:OF CONTEXT IL:WITH (OR (IL:LISTGET PROPS :PROFILE)
  (SAVE-PROFILE (COPY-PROFILE "READ-PRINT"))))
(IL:REPLACE (EDIT-CONTEXT EVAL-IN-PROCESS) IL:OF CONTEXT IL:WITH (OR (IL:LISTGET PROPS :EVAL-IN-PROCESS)
  (EVAL-IN-PROCESS)))
(IL:REPLACE (EDIT-CONTEXT EVAL-FN) IL:OF CONTEXT IL:WITH (OR (IL:LISTGET PROPS :EVAL-FN)
  (XCL::PROFILE-ENTRY-VALUE '*EVAL-FUNCTION*)))
(WHEN (IL:LISTGET PROPS :SELECT-STRUCTURE)
  (IL:REPLACE (EDIT-CONTEXT FIND-CANDIDATE) IL:OF CONTEXT IL:WITH (CONS (IL:LISTGET PROPS :SELECT-STRUCTURE)
    (OR (IL:LISTGET PROPS
      :SELECT-INSTANCE)
    1)))))
```

```
(DEFUN START-PROCESS (CONTEXT)
```

```
;; the context is ready. start the sedit process. the rest of the initialization will happen in the sedit process, and the completion-event will be notified (by
;; SEDIT1) when the sedit is initialized.
```

```
(LET ((NAME (IL:FETCH (EDIT-CONTEXT ICON-TITLE) IL:OF CONTEXT))
  (EVENT (IL:|fetch| (EDIT-CONTEXT COMPLETION-EVENT) IL:|of| CONTEXT)))
  (IL:ADD-PROCESS (LIST 'SEdit1 (IL:KWOTE CONTEXT))
    'IL:NAME
    (IF NAME
      (IL:CONCAT EDITOR-NAME " " NAME)
      EDITOR-NAME))
  (IL:|until| (EQ EVENT (IL:AWAIT.EVENT EVENT)))))
```

```
;; THESE CAN ALL BE NUKED WITH THE NEW EDIT INTERFACE AND A DETACHED TTY/EDITOR (WOZ 1/25/91)
```

```
(IL:PUTPROPS SEDIT IL:|Definition-for-EDITL| SEDITL)
```

```
(IL:PUTPROPS SEDIT IL:|Definition-for-EDITE| SEDITE)
```

```
(IL:PUTPROPS SEDIT IL:|Definition-for-EDITDATE| IL:TTY/EDITDATE)
```

```
(IL:DEFINEQ
```

```
(SEdit
```

```
(IL:LAMBDA (EXPR COMS ATOM TYPE IFCHANGEDFN OPTIONS)
```

```
; Edited 10-Jul-91 19:04 by jds
```

```
;; Convert call to EDITE into sedit format (structure props options). The completion-fn is determined based on TYPE, since the file manager isn't very
;; consistent about IL:PROPLST and IL:ALIST. Since EDITE is supposed to wait for completion, create a completion event which is notified by the
;; completion-fn. Also, if the top cons is changed, try to smash the completed structure into EXPR to provide eqness.
```

```
;; IDEALLY: this whole mess wouldn't exist- if il:putdef could handle il:proplst, etc, then completion could simply call putdef, not special case as it does
;; here.
```

```
(LET* ((EVENT (IL:CREATE.EVENT "SEdit Completion"))
  (NEW-EXPR)
```

```

(COMPLETION-FN (OR (AND IL:FILEPKGFLG
                    (IL:SELECTQ TYPE
                     (IL:PROPLST (LET ((OLD-PROPS (IL:APPEND (IL:GETPROPLIST ATOM))))
                                     #'(LAMBDA (CONTEXT STRUCTURE CHANGED?)
                                           (FUNCALL #'PROPS-COMPLETION CONTEXT STRUCTURE
                                                    CHANGED? ATOM IFCHANGEDFN OLD-PROPS)
                                           (SETQ NEW-EXPR STRUCTURE)
                                           (IL:NOTIFY.EVENT EVENT))))))
                    (IL:VARS (WHEN (IL:EQMEMB 'IL:ALIST (IL:GETPROP ATOM 'IL:VARTYPE))
                                   (LET ((OLD-VAL (IL:MAPCAR (IL:FUNCTION CAR)
                                                             (IL:EVALV ATOM))))
                                       #'(LAMBDA (CONTEXT STRUCTURE CHANGED?)
                                           (FUNCALL #'ALIST-COMPLETION CONTEXT
                                                    STRUCTURE CHANGED? ATOM IFCHANGEDFN
                                                    OLD-VAL)
                                           (SETQ NEW-EXPR STRUCTURE)
                                           (IL:NOTIFY.EVENT EVENT))))))
                    NIL))
  (AND ATOM TYPE #'(LAMBDA (CONTEXT STRUCTURE CHANGED?)
                            (FUNCALL #'COMPLETION CONTEXT STRUCTURE CHANGED? ATOM TYPE
                                     IFCHANGEDFN)
                            (SETQ NEW-EXPR STRUCTURE)
                            (IL:NOTIFY.EVENT EVENT))))
  #'(LAMBDA (CONTEXT STRUCTURE CHANGED?)
      (SETQ NEW-EXPR STRUCTURE)
      (IL:NOTIFY.EVENT EVENT))))
(ROOT-CHANGED-FN (IL:SELECTQ TYPE
                  (IL:PROPLST (LIST (IL:FUNCTION PROPLST-CHANGED)
                                     ATOM))
                  (IL:VARS (LIST (IL:FUNCTION VAR-CHANGED)
                                     ATOM))
                  (IL:FNS (LIST (IL:FUNCTION FN-CHANGED)
                                 ATOM))
                  NIL)))
(COND
  (COMS (IL:TTY/EDITE EXPR COMS ATOM TYPE IFCHANGEDFN OPTIONS))
  (T (WHEN (AND IL:FILEPKGFLG (OR IL:CLISPARRAY (PROGN (IL:CLISPTRAN (CONS)
                                                                    (CONS))
                                                                    IL:CLISPARRAY)))
          (IL:SELECTQ TYPE
            (IL:PROPLST (IL:|for| X IL:|in| (IL:GETPROPLIST ATOM) IL:|unless| (OR (IL:NLISTP X)
                                                                              (IL:GETHASH X
                                                                              IL:CLISPARRAY))
                                IL:|do| (IL:PUTHASH X (CONS (CAR X)
                                                            (CDR X))
                                IL:CLISPARRAY)))
            (IL:VARS (WHEN (IL:EQMEMB 'IL:ALIST (IL:GETPROP ATOM 'IL:VARTYPE))
                          (IL:|for| X IL:|in| (IL:EVALV ATOM) IL:|unless| (OR (IL:NLISTP X)
                                                                              (IL:GETHASH X IL:CLISPARRAY))
                                IL:|do| (IL:PUTHASH X (CONS (CAR X)
                                                            (CDR X))
                                IL:CLISPARRAY))))
            NIL))
  (SEdit EXPR (LIST :NAME ATOM :TYPE TYPE :COMPLETION-FN COMPLETION-FN :ROOT-CHANGED-FN
                    ROOT-CHANGED-FN)
    OPTIONS)
  (UNLESS (IL:EQMEMB :DONTWAIT OPTIONS)
    (IL:|until| (EQ EVENT (IL:AWAIT.EVENT EVENT))))
  ;; EDITE is for side effects (but we return the correct structure anyway. If the user replaced the top cons, smash the new structure
  ;; into it. Have to copy the new structure in this case because, if the user wrapped the top cons, smashing into it will result in a
  ;; circular list. Additionally, if there is an sedit root-changed-fn, assume the caller handled the root change then, and eqness is not
  ;; necessary.
  (IF (OR (EQ NEW-EXPR EXPR)
          ROOT-CHANGED-FN
          (NOT (CONSP EXPR))
          (NOT (CONSP NEW-EXPR)))
      NEW-EXPR
      (IL:RPLNODE2 EXPR (COPY-TREE NEW-EXPR))))))

```

(SEditL

```

(IL:LAMBDA (EDITEXPR EDITCOMS ATOM MESSAGE EDITCHANGES) ; Edited 25-Jan-91 13:45 by woz
  (DECLARE (SPECIAL TYPE))

```

;;; this is SEdit's definition for EDITL. if there are no COMS (normal case) we start an interactive SEdit. otherwise, we run the TTY editor to execute the
 ;;; coms, and arrange to start an SEdit if it stops for input.

```

(COND
  (EDITCOMS
    ;; used to push stuff on il:editmacros, now we bind il:tyeditfn
    ;; (il:resetvar il:editmacros (cons '(il:tyt: nil (il:e (tyfn il:atm type) t)) il:editmacros) (il:tyt/editl editexpr editcoms atom message
    ;; editchanges))
    (LET ((IL:TTYEDITFN #'(LAMBDA NIL (TTYFN ATOM TYPE))))
      (DECLARE (SPECIAL IL:TTYEDITFN))
      (IL:TTY/EDITL EDITEXPR EDITCOMS ATOM MESSAGE EDITCHANGES)))

```

```
(T (SEDLT (CAR EDITEXPR)
  (LIST :NAME ATOM :TYPE (AND (BOUNDP 'TYPE)
    TYPE)))
  EDITEXPR)))
```

(FN-CHANGED

```
(IL:LAMBDA (STRUCTURE ATOM) ; Edited 7-Jul-87 12:59 by DCB
  (COND
    ((NOT (IL:CCODEP (IL:GETD ATOM)))
      (IL:PUTD ATOM STRUCTURE))
    ((IL:LISTP (IL:GETPROP ATOM 'IL:EXPR))
      (IL:PUTPROP ATOM 'IL:EXPR STRUCTURE))
    (T (IL:SHOULDNT "where did this come from?")))))
```

(PROP-CHANGED

```
(IL:LAMBDA (STRUCTURE ATOM) ; Edited 7-Jul-87 12:59 by DCB
  (IL:PUTPROP ATOM 'IL:EXPR STRUCTURE)))
```

(PROPLST-CHANGED

```
(IL:LAMBDA (STRUCTURE ATOM) ; Edited 7-Jul-87 12:59 by DCB
  (IL:SETPROPLST ATOM STRUCTURE)))
```

(VAR-CHANGED

```
(IL:LAMBDA (STRUCTURE ATOM) ; Edited 7-Jul-87 12:59 by DCB
  (SET ATOM STRUCTURE)))
```

(ALIST-COMPLETION

```
(IL:LAMBDA (CONTEXT STRUCTURE CHANGED? ATOM IFCHANGEDFN OLD-KEYS) ; Edited 18-Jan-88 15:43 by woz
  (WHEN (EQ CHANGED? T)
    ;; don't do anything if changed is NIL or :ABORT
    (LET (FOUND-CHANGE OLD-VALUE)
      (IL:FOR X IL:IN OLD-KEYS IL:UNLESS (IL:ASSOC X STRUCTURE) IL:DO (IL:MARKASCHANGED (LIST ATOM X)
        'IL:ALISTS NIL)
        (IL:SETQ FOUND-CHANGE T))
      (IL:FOR X IL:IN STRUCTURE IL:WHEN (AND (IL:LISTP X)
        (NOT (AND IL:CLISPARRAY (IL:SETQ OLD-VALUE (IL:GETHASH X
          IL:CLISPARRAY
            ))
          (EQ (CAR X)
            (CAR OLD-VALUE))
          (EQ (CDR X)
            (CDR OLD-VALUE))))))
        IL:DO (IL:PUTHASH X NIL IL:CLISPARRAY)
          (IL:MARKASCHANGED (LIST ATOM (CAR X))
            'IL:ALISTS NIL)
          (IL:SETQ FOUND-CHANGE T))
      (WHEN (NOT FOUND-CHANGE)
        (COMPLETION CONTEXT STRUCTURE CHANGED? ATOM 'IL:ALISTS IFCHANGEDFN))))))
```

(COMPLETION

```
(IL:LAMBDA (CONTEXT STRUCTURE CHANGED? ATOM TYPE IFCHANGEDFN) ; Edited 3-Jan-92 14:11 by jrb:
  (IF (OR (NOT CHANGED?)
    (EQ CHANGED? :ABORT))
    NIL
    (PROGN (FIX-EDITDATE CONTEXT STRUCTURE TYPE)
      (COND
        ((EQ TYPE 'IL:FNS)
          (IL:PUTDEF ATOM TYPE STRUCTURE)
          ;; (if (CCODEP (GETD atom)) then (if (NEQ structure (GETPROP atom (QUOTE EXPR))) then (SHOULDNT 'where did this
          ;; come from?) else (if (OR (EQ DFNFLG (QUOTE PROP)) (EQ DFNFLG (QUOTE ALLPROP))) then (SETQ message ' NOT
          ;; unsaved.') else (UNSAVEDEF atom) (SETQ message ' unsaved.')) (if (OPENWP (fetch DisplayWindow of context)) then
          ;; (printout (get.prompt.window context) atom message) else (* ; 'window was closed. msg in promptwindow.') (printout
          ;; PROMPTWINDOW T atom message))) else (if (NEQ structure (GETD atom)) then (if (NULL (GETD atom)) then (PUTD atom
          ;; structure) else (SHOULDNT 'where did this come from?))))
        )
        (IFCHANGEDFN
          ;; this is a bit wrong: the doc for edit says the ifchangedfn gets called with the last arg NIL if the editor is aborted. But
          ;; we don't call the ifchangedfn at all if the user did an abort command. The idea is that ABORT is implemented as
          ;; "don't install even if changes we're made"
          (FUNCALL IFCHANGEDFN ATOM STRUCTURE TYPE T))
        ((IL:NEQ TYPE 'IL:PROPLST)
          (IL:MARKASCHANGED ATOM TYPE))))))
  (WHEN (AND (IL:LITATOM ATOM)
    IL:ADDSPELLFLG)
    (IL:ADDSPELL ATOM)))
```

(PROPS-COMPLETION

```
(IL:LAMBDA (CONTEXT STRUCTURE CHANGED? ATOM IFCHANGEDFN OLDPROPS)
; Edited 20-Apr-88 11:39 by woz
(WHEN (EQ CHANGED? T)
;; don't do anything if changed? is NIL or :ABORT
(IL:BIND OLD-VALUE FOUND-ONE IL:FOR NEW-PROP IL:ON (IL:GETPROPLIST ATOM) IL:BY (CDDR NEW-PROP)
IL:UNLESS (IL:FOR OLD-PROP IL:ON OLDPROPS IL:BY (CDDR OLD-PROP) IL:WHEN (EQ (CAR OLD-PROP)
(CAR NEW-PROP))
IL:DO (RETURN (AND (EQ (CADR OLD-PROP)
(CADR NEW-PROP))
(OR (IL:NLISTP (CADR OLD-PROP))
(AND IL:CLISPARRAY (IL:SETQ OLD-VALUE (IL:GETHASH (CADR NEW-PROP)
IL:CLISPARRAY))
(EQ (CAADR NEW-PROP)
(CAR OLD-VALUE))
(EQ (CDADR NEW-PROP)
(CDR OLD-VALUE))
(OR (IL:PUTHASH (CADR NEW-PROP)
NIL IL:CLISPARRAY)
T))))))
IL:DO (IL:MARKASCHANGED (LIST ATOM (CAR NEW-PROP))
'IL:PROPS NIL)
(IL:SETQ FOUND-ONE T))))
```

(TTYFN

```
(IL:LAMBDA (ATM TYPE)
; Edited 21-Jan-91 12:02 by woz
(DECLARE (SPECIAL IL:L IL:EDITCHANGES))
;; this is a replacement for the TTY editor's TTY: command, which starts an SEdit process to do interactive editing for a while. it uses the TTY
;; editor's edit chain to determine the initial selection in the structure, and scrolls the window to make sure the selection's visible. it then waits until
;; the user signals that they've done enough editing (usually by closing or shrinking the window)
(LET* ((EDIT-CHANGES IL:EDITCHANGES)
(EVENT (IL:CREATE.EVENT "SEdit TTYFN Completion"))
(COMPLETION-FN #'(LAMBDA (CONTEXT STRUCTURE CHANGED?)
(WHEN (EQ CHANGED? T)
(RPLACA (CDR EDIT-CHANGES)
T))
(IL:NOTIFY.EVENT EVENT))))
(CONTEXT (SEdit (CAR (LAST IL:L))
(LIST :NAME ATM :TYPE TYPE :COMPLETION-FN COMPLETION-FN)))
NODE)
(IL:WITH-MONITOR (IL:|fetch| CONTEXT-LOCK IL:|of| CONTEXT)
(WHEN (IL:SETQ NODE (LOCATE-NODE-FROM-EDITCHAIN IL:L (IL:|fetch| ROOT IL:|of| CONTEXT)))
(SELECTION-DOWN CONTEXT)
(SELECT-NODE CONTEXT NODE)
(SET-POINT-NOWHERE (IL:|fetch| CARET-POINT IL:|of| CONTEXT))
(NORMALIZE-SELECTION CONTEXT)
(SELECTION-UP CONTEXT)))
;; let the user do their editing, then signal completion, before we return
(IL:|until| (EQ EVENT (IL:AWAIT.EVENT EVENT))))))
```

(LOCATE-NODE-FROM-EDITCHAIN

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

;; when SEdit is called under the TTY editor, it gets an edit chain to determine the initial selection. this process finds the node that editchain refers to (or
;; returns NIL if no such node exists)

```
(IF (NULL CHAIN)
ROOT
(IL:FOR SUBNODE IL:IN (CDR (IL:FETCH SUB-NODES IL:OF (LOCATE-NODE-FROM-EDITCHAIN (CDR CHAIN)
ROOT)))
IL:THEREIS (EQ (IL:FETCH STRUCTURE IL:OF SUBNODE)
(CAR CHAIN))))))
)
```

;; SEdit's hack way of fixing edit dates

```
(DEFUN FIX-EDITDATE (CONTEXT STRUCTURE TYPE)
```

```
; Edited 3-Jan-92 14:24 by jrb:
```

```
;; If we can find an :EDITDATE-OFFSET, look for an editdate comment at that offset, and smash or insert one appropriately
(LET ((OFFSET (COND
((EQ TYPE 'FNS)
2)
((GET TYPE :DEFINED-BY)
(GET (CAR STRUCTURE)
:EDITDATE-OFFSET)))))
NEWDATESTRING)
(WHEN (AND IL:INITIALS OFFSET)
(SETQ NEWDATESTRING (IL:CONCAT "Edited " (IL:DATE (IL:DATEFORMAT IL:NO.SECONDS))
" by " IL:INITIALS))
(IF (IL:EDITDATE? (NTH OFFSET STRUCTURE))
```

```

(LET ((EDITDATE-NODE (SUBNODE (1+ OFFSET)
                                (SUBNODE 1 (IL:FETCH ROOT IL:OF CONTEXT))))))
  (REPLACE-NODE CONTEXT EDITDATE-NODE (PARSE-NEW `
                                                    (IL:* IL:; (IL:\\, NEWDATESTRING)
                                                    CONTEXT)))
  (LET ((POINT (IL:CREATE EDIT-POINT
                            POINT-NODE IL:_ (SUBNODE 1 (IL:FETCH ROOT IL:OF CONTEXT))
                            POINT-INDEX IL:_ OFFSET
                            POINT-TYPE IL:_ 'STRUCTURE)))
    (INSERT POINT CONTEXT (PARSE-NEW `
                                    (IL:* IL:; (IL:\\, NEWDATESTRING)
                                    CONTEXT))))
  ;; This looks bogus, BUT: we just changed the structure in mid-editor-shutdown, so we need to tell SEdit that we really didn't mean it,
  ;; sort of...
  (THROW-AWAY-CHANGES CONTEXT)))

```

;; Mess around with the tty editor's TTY: command by defining a hook and then making TTY: a macro which calls the hook.

```
(DEFUN SMART-TTYFN ())
```

;;; This is a replacement for the TTY editor's TTY: command, which is supposed to start up a TTY editor. We first check to see if we're

```

(DECLARE (SPECIAL IL:L IL:TTYEDITFN))
(IF (AND (BOUNDP 'IL:TTYEDITFN)
         IL:TTYEDITFN)
    (FUNCALL IL:TTYEDITFN)
    (IL:EDITLO IL:L NIL 'IL:TTY\; 'IL:TTY\;)))

```

```

(PUSHNEW '(IL:TTY\; NIL (IL:E (SMART-TTYFN
                               T)))
  IL:EDITMACROS :TEST #'IL:EQUAL)

```

```
(IL:DEFINEQ
```

```
(PRETTY-PRINT
```

```
(IL:LAMBDA (STRUCTURE STREAM RIGHT-MARGIN)
```

; Edited 7-Jul-87 12:59 by DCB

;;; with just a little hacking, SEdit can be used to prettyprint functions onto TEdit streams. we make up a slightly weird context, and run the parser and
 ;; linearizer each once. stream is actually the textobj of the tedit stream. note that right.margin is in micas (since that's the unit that interpress font
 ;; widths are expressed in)

```

(OR (BOUNDP 'PRETTY-PRINT-ENV)
    (CREATE-PRETTY-PRINT-ENV))
(LET ((CONTEXT (IL:CREATE EDIT-CONTEXT
                          DISPLAY-WINDOW IL:_ STREAM
                          ENVIRONMENT IL:_ PRETTY-PRINT-ENV
                          CURRENT-X IL:_ 0
                          COMMENT-WIDTH IL:_ (IL:FIXR (IL:TIMES 200 IL:MICASPERPT))
                          COMMENT-SEPARATION IL:_ (IL:FIXR (IL:TIMES 30 IL:MICASPERPT))))
  (ROOT (IL:CREATE EDIT-NODE
                  NODE-TYPE IL:_ TYPE-ROOT
                  SUB-NODES IL:_ (LIST 0)
                  START-X IL:_ 0
                  DEPTH IL:_ 0)))
  (IL:REPLACE CURRENT-NODE IL:OF CONTEXT IL:WITH ROOT)
  (PARSE STRUCTURE CONTEXT)
  (COMPUTE-ALL-FORMATS CONTEXT NIL)
  (LINEARIZE (SUBNODE 1 ROOT)
    CONTEXT
    (IL:FIXR RIGHT-MARGIN))))

```

```
(MAP-FONT
```

```
(IL:LAMBDA (FONT ENV)
```

; Edited 17-Nov-87 10:43 by DCB

;; this is called when using the prettyprint environment, under output.string. we have to map the font into something acceptable to TEDIT.INSERT
 ;; (since interpress fonts confuse it)

```

(COND
  ((EQ FONT (IL:FETCH DEFAULT-FONT IL:OF ENV))
   IL:DEFAULTFONT)
  ((EQ FONT (IL:FETCH KEYWORD-FONT IL:OF ENV))
   IL:CLISPFONT)
  ((EQ FONT (IL:FETCH ITALIC-FONT IL:OF ENV))
   IL:ITALICFONT)
  ((EQ FONT (IL:FETCH COMMENT-FONT IL:OF ENV))
   IL:COMMENTFONT)
  ((EQ FONT (IL:FETCH BROKEN-ATOM-FONT IL:OF ENV))
   IL:BOLDFONT)
  (T (IL:SHOULDNT "unexpected font!"))))

```

```
)
```

;; these guys allow you to print and read structures with broken atoms and gaps. just a convenience for the loser who forgets to get them out of his
 ;; code.


```
(DEFUN MAKE-BROKEN-ATOM (STRING)
  (IL:create| BROKEN-ATOM
    ATOM-CHARS IL:_ STRING))
```

```
(DEFUN PRINT-BROKEN-ATOM (BROKEN-ATOM STREAM X)
  (FORMAT STREAM "#. (~S ~S)" 'MAKE-BROKEN-ATOM (IL:fetch| ATOM-CHARS IL:of| BROKEN-ATOM)
  T)
```

```
(DEFUN MAKE-GAP (ITEM)
  (IL:create| GAP
    LINEAR-ITEM IL:_ ITEM))
```

```
(DEFUN PRINT-GAP (GAP STREAM X)
  (FORMAT STREAM "#. (~S '~S)" 'MAKE-GAP (IL:fetch| LINEAR-ITEM IL:of| GAP))
  T)
```

```
(IL:DEFPRINT 'BROKEN-ATOM 'PRINT-BROKEN-ATOM)
```

```
(IL:DEFPRINT 'GAP 'PRINT-GAP)
```

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

FUNCTION INDEX

ALIST-COMPLETION	6	MAKE-GAP	9	RESET	2
AWAKE-COMMAND-PROCESS	3	MAP-FONT	8	SAVE-WINDOW-REGION	2
AWAKE-ME	3	MARKASCHANGEDFN	3	SEdit	1
COMPLETION	6	NEW-FUNCTION-BODY	3	SEdITE	4
DISINTEGRATE-CONTEXT	3	PRETTY-PRINT	8	SEditL	5
FIX-EDITDATE	7	PRINT-BROKEN-ATOM	9	SET-OPTIONS	4
FN-CHANGED	6	PRINT-GAP	9	SET-PROPS	4
GET-CONTEXT	2	PROP-CHANGED	6	SMART-TTYFN	8
GET-WINDOW-REGION	2	PROPLST-CHANGED	6	START-PROCESS	4
LOCATE-NODE-FROM-EDITCHAIN	7	PROPS-COMPLETION	7	TTYFN	7
MAKE-BROKEN-ATOM	9	QUERY-THROW-AWAY-CHANGES	4	VAR-CHANGED	6

VARIABLE INDEX

IL:*DISPLAY-EDITOR*	1	CONTEXTS	1	REGIONS	1
---------------------------	---	----------------	---	---------------	---

PROPERTY INDEX

SEdit	4	IL:SEdit-TOPLEVEL	1
-------------	---	-------------------------	---
