File created: 16-May-90 18:18:52 **{DSK}<usr>local>lde>lispcore>sources>IL-ERROR-STUFF.;2**

changes to: (VARS IL-ERROR-STUFFCOMS)

previous date: 2-Mar-88 16:39:33 {DSK}<usr>local>lde>lispcore>sources>IL-ERROR-STUFF.;1

Read Table: XCL

Package: INTERLISP

Format: XCCS


```
(RPAQQ IL-ERROR-STUFFCOMS
       ((FNS HELP SHOULDNT ERROR ERRORX INTERRUPT FAULTEVAL FAULTAPPLY OLDFAULT1 ERRORMESS ERRORMESS1
             SMARTARGLIST \\SIMPLIFY.CL.ARGLIST)
        (INITVARS (HELPDEPTH 7)
               (BREAKDELIMITER "
                      ")
               (HELPTIME 1000)
               (HELPCLOCK)
               (NLSETQGAG T)
               (*MACROEXPAND-HOOK* 'CL:FUNCALL)
               (COMPILERMACROPROPS '(DMACRO ALTOMACRO BYTEMACRO MACRO)))
        ;; for backward compatibility. Used currently by window mouse handler
        (VARS (NBREAKS 0))
        (DECLARE\: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                              (NLAML)
                                                                              (LAMA ERRORX)))))

(DEFINEQ

(HELP
  (LAMBDA (MESS1 MESS2)                                          ; Edited  5-Feb-88 14:38 by amd
    "Used to signify 'I don't know what I should do'"
    (CONDITIONS:INVOKE-DEBUGGER (MAKE-CONDITION 'SIMPLE-CONDITION :FORMAT-STRING "Help!~@{~@[ ~A~]~}"
                                       :FORMAT-ARGUMENTS (LIST MESS1 MESS2)))))

(SHOULDNT
  (LAMBDA (MESS)                                                 ; Edited  9-Mar-87 11:41 by jrb:
    (HELP "Shouldn't happen:" MESS)))

(ERROR
  (LAMBDA (MESS1 MESS2 NOBREAK)                                  ; Edited 18-Jan-88 19:05 by amd
    (DECLARE (GLOBALVARS NLSETQGAG))
    (COND
       ((AND NOBREAK (NEQ HELPFLAG 'BREAK!))                     ; An ERROR! cum message.
        (SIGNAL (ERRM-TO-CONDITION 17 (CONS MESS1 MESS2)))
        (ERRORMESS1 MESS1 MESS2)
        (ERROR!))
       (T (SETERRORN 17 (CONS MESS1 MESS2))
          (ERRORX *LAST-CONDITION*)))))

(ERRORX
  (CL:LAMBDA (&OPTIONAL ERXM)                                    ; Edited  7-Apr-87 18:12 by amd
         (AND ERXM (NOT (TYPEP ERXM 'CONDITION))
              (SETERRORN (CAR ERXM)
                     (CADR ERXM)))
         (RESETLST
             (LET ((ERRORPOS (FIND-DEBUGGER-ENTRY-FRAME 'ERRORX T)))
                  (DECLARE (CL:SPECIAL ERRORPOS))
                  (RESETSAVE NIL (LIST 'RELSTK ERRORPOS))
                  (PROCEED-CASE (CL:ERROR *LAST-CONDITION*)
                         (PROCEED (CONDITION)
                                :REPORT "Retry execution" (ENVAPPLY (STKNAME ERRORPOS)
                                                                    (STKARGS ERRORPOS)
                                                                    (STKNTH 1 ERRORPOS ERRORPOS)
                                                                    ERRORPOS T T)))))))

(INTERRUPT
  (LAMBDA (INTFN)                                                ; Edited 24-Nov-86 19:16 by amd
    (DEBUGGER :CONDITION (MAKE-CONDITION 'SI::INTERRUPT :FUNCTION INTFN))))

(FAULTEVAL
  (LAMBDA (FAULTX)
    (DECLARE (LOCALVARS . T))                                    ; Edited 12-Mar-87 09:56 by jds
    (PROG (TEM TEM2)
          (COND
             ((LISTP FAULTX)
              (COND
                 ((LITATOM (SETQ TEM (CAR FAULTX)))
```

```
                (COND
                   ((AND (SETQ TEM2 (GETMACROPROP TEM COMPILERMACROPROPS))
                          (NOT (EQUAL FAULTX (SETQ TEM2 (MACROEXPANSION FAULTX TEM2)))))
                    (COND
                       (CLISPARRAY (PUTHASH FAULTX (OR (LISTP TEM2)
                                                       (LIST 'PROGN TEM2))
                                                   CLISPARRAY)))
                    (RETURN (\\EVAL TEM2)))
                   ((SETQ TEM2 (GET TEM 'MACRO-FN))
                    (RETURN (\\EVAL (CL:FUNCALL *MACROEXPAND-HOOK* TEM2 FAULTX NIL))))
                   (T (SETQ TEM (GETD (CAR FAULTX)))))))
             (COND
                ((AND (LISTP TEM)
                      (FMEMB (CAR TEM)
                             LAMBDASPLST))
                 (RETURN (\\EVALFORMASLAMBDA FAULTX))))))
          (RETURN (COND
                     ((AND DWIMFLG (GETD 'NEWFAULT1))
                      (NEWFAULT1 FAULTX))
                     (T (OLDFAULT1 FAULTX)))))))
```

(FAULTAPPLY

```
(LAMBDA (FAULTFN FAULTARGS)                                    ; Edited 24-Feb-87 14:26 by amd
  (COND
     ((CCODEP FAULTFN)
```

;; hack to handle case where microcode doesn't know how to FUNCALL a CCODEP

```
      (SPREADAPPLY '\\INTERPRETER-DUMMY (UNINTERRUPTABLY
                                          (PUTD '\\INTERPRETER-DUMMY FAULTFN T)
                                          FAULTARGS)))
     (T (PROG ((DEF (COND
                       ((CL:SYMBOLP FAULTFN)
                        (GETD FAULTFN))
                       (T FAULTFN)))
               %LEXICAL-ENVIRONMENT%)
          RETRY
              (COND
                 ((TYPEP DEF 'CLOSURE)                         ; an interpreted closure
                  (SETQ %LEXICAL-ENVIRONMENT% (CLOSURE-ENVIRONMENT DEF))
                  (SETQ DEF (CLOSURE-FUNCTION DEF))
                  (GO RETRY)))
              (RETURN (PROG (TRAN TRANFN)
                            (OR (AND (LISTP DEF)
                                     (FMEMB (CAR DEF)
                                            LAMBDASPLST))
                                (GO OUT))
                            (COND
                               ((OR (SETQ TRAN (GETHASH DEF CLISPARRAY))
                                    (AND (SETQ TRANFN (CAR (CDR (ASSOC (CAR DEF)
                                                                       LAMBDATRANFNS))))
                                         (LISTP (SETQ TRAN (CL:FUNCALL TRANFN DEF)))
                                         (PROGN (AND CLISPARRAY (PUTHASH DEF TRAN CLISPARRAY))
                                                T)))
```

;; either in CLISPARRAY or translated by lambda-tran

```
                                (SETQ DEF TRAN))
                               (%LEXICAL-ENVIRONMENT%
```

;; found the environment

```
                                )
                               (T (GO OUT)))
                            (RETURN (APPLY DEF FAULTARGS))
                        OUT (RETURN (COND
                                       ((AND DWIMFLG (GETD 'NEWFAULT1))
                                        (NEWFAULT1 FAULTFN FAULTARGS T))
                                       (T (OLDFAULT1 FAULTFN FAULTARGS T)))))))))))
```

(OLDFAULT1

```
(LAMBDA (FAULTX FAULTARGS FAULTAPPLYFLG)                       (* |lmm| " 6-Nov-86 02:10")
  (CL:CERROR "Re-execute" (COND
                             (FAULTAPPLYFLG (MAKE-CONDITION 'UNDEFINED-FUNCTION-IN-APPLY :NAME FAULTX
                                                            :ARGUMENTS FAULTARGS))
                             ((NLISTP FAULTX)
                              (MAKE-CONDITION 'UNBOUND-VARIABLE :NAME FAULTX))
                             (T (MAKE-CONDITION 'UNDEFINED-FUNCTION :NAME (CAR FAULTX)))))
  (COND
     (FAULTAPPLYFLG (CL:APPLY FAULTX FAULTARGS))
     (T (EVAL FAULTX))))
```

(ERRORMESS

```
(LAMBDA (U)                                                    ; Edited 24-Nov-86 13:46 by amd
```

;; Replaces ERRORM.

;; merged FAULT2 printing in, driven off of extra information on ERRORN - rrb 7/83

```
    (LET ((CONDITION (COND
                        ((NULL U)
                         *LAST-CONDITION*)
                        ((TYPEP U 'CONDITION)
                         U)
                        (T (ERRM-TO-CONDITION (CAR U)
                                  (CADR U))))))
         (COND
            ((AND LISPXHISTORY (NOT (TYPEP CONDITION 'STORAGE-CONDITION)))
             (LISPXPUT '*ERROR* (CL:TYPE-OF CONDITION))))        (* "Used to be:" (COND (LISPXPRINTFLG
                                                                 (PROG ((EXTRAMES (CADDR U)))
                                                                 (LISPXTERPRI T) (LISPXPRIN1
                                                                 (ERRORSTRING (CAR U)) T) (LISPXTERPRI T)
                                                                 (LISPXPRIN2 (CADR U) T) (COND
                                                                 ((LISTP EXTRAMES) (* |;;| "in top level unbound atoms this is
                                                                 the litatom NORMAL for which nothing should print.")
                                                                 (LISPXPRIN1 (QUOTE " {in ") T)
                                                                 (LISPXPRIN2 (CAR EXTRAMES) T T)
                                                                 (LISPXPRIN1 (QUOTE "}") T) (COND
                                                                 ((CDR EXTRAMES) (LISPXPRIN1
                                                                 (QUOTE " in ") T) (LISPXPRIN2 (CDR EXTRAMES) T T)))))
                                                                 (LISPXTERPRI T))) (T (ERRORM U))))
         (CL:PRINC CONDITION *ERROR-OUTPUT*))))
```

(**ERRORMESS1**

```
  (LAMBDA (MESS1 MESS2 MESS3)                                    ; Edited  2-Mar-88 16:36 by amd
```

;; Prints messages for help and error

```
    (PROG (BADGUY MESSAGE)
          (COND
             ((AND (NULL MESS1)
                   (NULL MESS2))
              (PRINT MESS3 *STANDARD-OUTPUT* T)
              (RETURN)))
          (PRIN1 MESS1 *STANDARD-OUTPUT*)
          (COND
             ((OR (ATOM MESS1)
                  (STRINGP MESS2))
              (SPACES 1 *STANDARD-OUTPUT*))
             (T (TERPRI *STANDARD-OUTPUT*)))
          (COND
             ((STRINGP MESS2)
              (PRIN1 MESS2 *STANDARD-OUTPUT*)
              (TERPRI *STANDARD-OUTPUT*))
             (T (PRINT MESS2 *STANDARD-OUTPUT* T))))))
```

(**SMARTARGLIST**

```
  (LAMBDA (FN EXPLAINFLG TAIL)                                   ; Edited 15-Jan-88 18:34 by bvm:
```

;; Provide the argument list for the function named FN. FN may also be a macro, in which case its arg list is fake, but returned anyhow.

;; FN can also be the actual (LAMBDA --) form.

```
    (PROG (TEM DEF OTHERDEF)
          (COND
             ((NOT (LITATOM FN))                                 ; Got a lambda form to return the arglist for
              (RETURN (COND
                         ((AND EXPLAINFLG (LISTP FN)
                               (EQ (CAR FN)
                                   'CL:LAMBDA))                  ; We're in EXPLAIN mode, and it was a common lisp function, so
                                                                 ; hack up the CL arg list
                          (\\SIMPLIFY.CL.ARGLIST (CADR FN)))
                         (T                                      ; Otherwise, just return the conventional arg list
                            (ARGLIST FN))))))
      RETRY
          (COND
             ((SETQ TEM (GET FN 'BROKEN))
```

;; It's a broken function, so get the REAL function's arg list & return it:

```
              (RETURN (SMARTARGLIST TEM EXPLAINFLG)))
             ((SETQ TEM (GETLIS FN '(ARGNAMES)))
```

;; gives user an override. Also provides a way of supplying args for nospread fns, whose Interlisp "arglist" is uninteresting, or whose
;; Common Lisp arglist would otherwise vanish upon being compiled by the Interlisp compiler. We use GETLIS rather than GET, so
;; that user can force the arglist to appear to be NIL, even if it isn't really.

;; Arg names are used for two purposes: explaining, as with ?=, and breaking/advising. For nospread functions, want the latter to be
;; a legitimate arglist. The format used for this is (NIL explainForm . validForm). Note that this alternative form is unambiguous, since
;; NIL can never be an argument name

```
              (RETURN (COND
                         ((OR (NLISTP (SETQ TEM (CADR TEM)))
                              (NOT (NULL (CAR TEM))))
                          TEM)
                         (EXPLAINFLG (CADR TEM))
                         (T (CDDR TEM)))))
             ((EXPRP (SETQ DEF (OR (GET FN 'ADVISED)
                                   (GETD FN)
```

```
                                (GET FN 'EXPR))))                    ; Have an interpreted def lying around
            (SELECTQ (CAR (LISTP DEF))
                 ((NLAMBDA LAMBDA)                                   ; Nice Interlisp forms--return args directly
                     (RETURN (CADR DEF)))
                 (CL:LAMBDA                                          ; Can handle these if explaining, but first simplify
                           (CL:WHEN EXPLAINFLG
                               (RETURN (\\SIMPLIFY.CL.ARGLIST (CADR DEF)))))
                 NIL))
          ((AND EXPLAINFLG (SETQ OTHERDEF (GETDEF FN 'FUNCTIONS 'CURRENT '(NOERROR NOCOPY)))
                (SELECTQ (CAR OTHERDEF)
                     ((DEFMACRO CL:DEFUN)
                         T)
                     ((DEFDEFINER DEFCOMMAND)
                          (|pop| OTHERDEF))
                     NIL))
           ;; The FN (or macro) has an in-core source definition.  Gather the arg list from that, as most authoritative:
           (RETURN (\\SIMPLIFY.CL.ARGLIST (CL:THIRD (REMOVE-COMMENTS OTHERDEF))))))
        (COND
           ((AND (OR DEF (SETQ DEF (GET FN 'CODE)))
                 (OR (EXPRP DEF)
                     (CCODEP DEF)))
            ;; The function has a definition in force, or there's a back-up compiled or source definition that's worth consulting.
            (COND
               ((OR (NOT EXPLAINFLG)
                    (NEQ (ARGTYPE DEF)
                         3))                                         ; If EXPLAINFLG is true and we have an NLAMBDA*, then there
                                                                     ; might be a macro with a more interesting arg list--hold off.
                (RETURN (ARGLIST DEF EXPLAINFLG)))
               ((AND (CCODEP DEF)
                     (LISTP (SETQ TEM (ARGLIST DEF EXPLAINFLG))))
                                                                     ; Had an interesting stored arglist.
                (RETURN TEM)))))
        (RETURN (COND
                   ((AND EXPLAINFLG (SETQ TEM (GETMACROPROP FN COMPILERMACROPROPS)))
                                                                     ; If we're explaining, then we might be able to get an interesting
                                                                     ; arg list out of a macro
                    (SELECTQ (CAR (LISTP TEM))
                         ((LAMBDA NLAMBDA OPENLAMBDA)                ; These have conventional args in second position
                             (CADR TEM))
                         (=                                          ; Get args for synonym
                            (SMARTARGLIST (CDR TEM)
                                   EXPLAINFLG))
                         (COND
                            ((LISTP (SETQ TEM (CAR TEM)))            ; Substitution macro--TEM is now the arg list
                             (COND
                                ((CDR (LAST TEM))                    ; Last element is a tail.  Turn (A B . C) into (A B ... C).  Following
                                                                     ; depends on APPEND working this way on improper lists
                                 (APPEND TEM (LIST '|...| (CDR (LAST TEM)))))
                                (T TEM)))
                            (DEF                                     ; No args, or computed macro (either of the form CompilerFN or
                                                                     ; (args . expansionFn)).  Fall back on the definition above
                                 (ARGLIST DEF EXPLAINFLG)))))
                   ((AND (NOT DEF)
                         (SETQ TEM (FNCHECK FN T NIL T TAIL))
                         (NEQ TEM FN))
                    (SETQ FN TEM)
                    (GO RETRY))
                   (T (ARGLIST FN EXPLAINFLG))))))
```

(\\SIMPLIFY.CL.ARGLIST

```
  (LAMBDA (LST)                                                      (* |lmm| "28-Sep-86 12:07")
    (|bind| SECTION |for| X |in| LST |collect| (CL:IF (LISTP X)
                                          (CASE SECTION
                                              (&OPTIONAL (CAR X))
                                              (&KEY (COND
                                                        ((LISTP (CAR X))
                                                         (CAAR X))
                                                        (T (CAR X))))
                                              (T                     ; assume destructuring
                                                 X))
                                          (CASE X
                                              (&AUX (GO $$OUT))
                                              ((&AUX &OPTIONAL &KEY &BODY &REST &ENVIRONMENT) (SETQ SECTION X))
                                              (T X))))))

)

(RPAQ? HELPDEPTH 7)

(RPAQ? BREAKDELIMITER "
                      ")

(RPAQ? HELPTIME 1000)
```

```
(RPAQ? HELPCLOCK )

(RPAQ? NLSETQGAG T)

(RPAQ? *MACROEXPAND-HOOK* 'CL:FUNCALL)

(RPAQ? COMPILERMACROPROPS '(DMACRO ALTOMACRO BYTEMACRO MACRO))
```

;; for backward compatibility. Used currently by window mouse handler

```
(RPAQQ NBREAKS 0)

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

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA ERRORX)
)

(RPAQQ IL-ERROR-STUFFCOMS
       ((FNS HELP SHOULDNT ERROR ERRORX INTERRUPT FAULTEVAL FAULTAPPLY OLDFAULT1 ERRORMESS ERRORMESS1
             SMARTARGLIST \\SIMPLIFY.CL.ARGLIST)
        (INITVARS (HELPDEPTH 7)
               (BREAKDELIMITER "
                      ")
               (HELPTIME 1000)
               (HELPCLOCK)
               (NLSETQGAG T)
               (*MACROEXPAND-HOOK* 'CL:FUNCALL)
               (COMPILERMACROPROPS '(DMACRO ALTOMACRO BYTEMACRO MACRO)))
        ;; for backward compatibility. Used currently by window mouse handler
        (VARS (NBREAKS 0))
        (DECLARE\: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                              (NLAML)
                                                                              (LAMA)))))

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

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA )
)

(PUTPROPS IL-ERROR-STUFF COPYRIGHT ("Venue & Xerox Corporation" 1987 1988 1990))
```

FUNCTION INDEX

VARIABLE INDEX