File created: 12-Sep-2022 21:16:02 **{DSK}<users>kaplan>local>medley3.5>working-medley>sources>BREAK-AND-TRACE.;2**

previous date: 16-May-90 12:12:42 {DSK}<users>kaplan>local>medley3.5>working-medley>sources>BREAK-AND-TRACE.;1

Read Table: XCL

Package: SYSTEM

Format: XCCS


```
(IL:RPAQQ IL:BREAK-AND-TRACECOMS
          (
```

;;; Support for tracing.

```
           (IL:VARIABLES XCL:*TRACE-DEPTH* XCL::*TRACED-FNS* IL:TRACEREGION)
           (IL:FUNCTIONS XCL:CREATE-TRACE-WINDOW)
           (IL:FUNCTIONS CREATE-TRACED-DEFINITION CONSTRUCT-ENTRY-PRINTING-CODE PRINT-TRACE-ENTRY-INFO
                  PRINT-TRACE-EXIT-INFO PRINT-TRACED-ARGUMENT PRINT-TRACED-CL-ARGLIST)
           (IL:VARIABLES XCL:*TRACE-LEVEL* XCL:*TRACE-LENGTH* XCL:*TRACE-VERBOSE* *TRACE-OUTPUT*)
           (IL:FNS TRACE UNTRACE)
           (IL:FUNCTIONS XCL:TRACE-FUNCTION)
```

;;; Support for breaking.

```
           (IL:FUNCTIONS XCL:BREAK-FUNCTION XCL:UNBREAK-FUNCTION XCL:REBREAK-FUNCTION CREATE-BROKEN-DEFINITION
                  UNBREAK-FROM-RESTORE-CALLS FINISH-UNBREAKING)
           (IL:VARIABLES IL:BROKENFNS XCL::*BREAK-HASH-TABLE* XCL::*UNBROKEN-FNS*)
           (IL:PROP IL:PROPTYPE IL:BROKEN)
```

;; The old Interlisp interface to breaking.

```
           (IL:FNS IL:BREAK IL:BREAK0 IL:REBREAK XCL:UNBREAK IL:UNBREAK0)
           (IL:FNS IL:BREAK1)
           (IL:SPECIAL-FORMS IL:BREAK1)
           (XCL:OPTIMIZERS IL:BREAK1)
```

;; Arrange for the proper compiler and package

```
           (IL:PROP (IL:FILETYPE IL:MAKEFILE-ENVIRONMENT)
                  IL:BREAK-AND-TRACE)
           (IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY IL:COMPILERVARS
                  (IL:ADDVARS (IL:NLAMA XCL:UNBREAK IL:REBREAK IL:BREAK UNTRACE TRACE)
                         (IL:NLAML IL:BREAK1)
                         (IL:LAMA)))))
```

;;; Support for tracing.

```
(DEFVAR XCL:*TRACE-DEPTH* 0)
```

```
(DEFVAR XCL::*TRACED-FNS* NIL
```

;;; A subset of the entries on IL:BROKENFNS, being those that resulted from calls to TRACE as opposed to calls to BREAK-FUNCTION.

```
   )
```

```
(DEFVAR IL:TRACEREGION (IL:|create| IL:REGION
                                    IL:LEFT IL:_ 8
                                    IL:BOTTOM IL:_ 3
                                    IL:WIDTH IL:_ 547
                                    IL:HEIGHT IL:_ 310))
```

```
(DEFUN XCL:CREATE-TRACE-WINDOW (&KEY (XCL::REGION IL:TRACEREGION)
                                     (XCL::OPEN? NIL)
                                     (XCL::TITLE "*Trace-Output*"))
```

;;; Create and return a window suitable for use as the value of *TRACE-OUTPUT*.

;;; REGION is the initial region of the window. It defaults to the value of IL:TRACEREGION.

;;; OPEN? is true if the newly-created window should be opened on the screen immediately. If false, the window will open the first time any output is
;;; sent to it.

```
   (LET ((XCL::WINDOW (IL:CREATEW XCL::REGION XCL::TITLE NIL (NOT XCL::OPEN?))))
        (IL:DSPSCROLL 'IL:ON XCL::WINDOW)
        XCL::WINDOW))
```

```
(DEFUN CREATE-TRACED-DEFINITION (TRACED-FN IN-FN FN-TO-CALL)
   (MULTIPLE-VALUE-BIND (LAMBDA-CAR ARG-LIST CALLING-FORM)
       (FUNCTION-WRAPPER-INFO TRACED-FN FN-TO-CALL)
```

```
     `(,LAMBDA-CAR ,(IF (EQ LAMBDA-CAR 'LAMBDA)
                        '(&REST XCL:ARGLIST)
                        ARG-LIST)
             ,@(AND ARG-LIST (MEMBER LAMBDA-CAR '(IL:LAMBDA IL:NLAMBDA))
                    `((DECLARE (SPECIAL ,@(IF (SYMBOLP ARG-LIST)
                                              (LIST ARG-LIST)
                                              ARG-LIST)))))
             (IL:\\CALLME ':TRACED ,(IF (NULL IN-FN)
                                        TRACED-FN
                                        `(,TRACED-FN :IN ,IN-FN))))
             (LET* (($THE-REAL-TRACE-OUTPUT$ (XCL:FOLLOW-SYNONYM-STREAMS (IL:\\GETSTREAM *TRACE-OUTPUT*)))
                    ($IMAGE-STREAM?$ (IL:IMAGESTREAMP $THE-REAL-TRACE-OUTPUT$)))
                   (LET ((*STANDARD-OUTPUT* $THE-REAL-TRACE-OUTPUT$)
                         (IL:FONTCHANGEFLG $IMAGE-STREAM?$))
                        (DECLARE (SPECIAL IL:FONTCHANGEFLG))
                        ,@(CONSTRUCT-ENTRY-PRINTING-CODE TRACED-FN IN-FN LAMBDA-CAR ARG-LIST))
                   (LET (($TRACED-FN-VALUES$ (MULTIPLE-VALUE-LIST (LET ((XCL:*TRACE-DEPTH* (1+ XCL:*TRACE-DEPTH*
                                                                                            )))
                                                                    ,CALLING-FORM))))
                        (LET ((*STANDARD-OUTPUT* $THE-REAL-TRACE-OUTPUT$)
                              (IL:FONTCHANGEFLG $IMAGE-STREAM?$))
                             (DECLARE (SPECIAL IL:FONTCHANGEFLG))
                             (PRINT-TRACE-EXIT-INFO ',TRACED-FN ',IN-FN $TRACED-FN-VALUES$))
                        (VALUES-LIST $TRACED-FN-VALUES$))))))

(DEFUN CONSTRUCT-ENTRY-PRINTING-CODE (TRACED-FN IN-FN LAMBDA-CAR ARG-LIST)
   `((PRINT-TRACE-ENTRY-INFO ',TRACED-FN ',IN-FN)
     (LET ((*PRINT-LEVEL* XCL:*TRACE-LEVEL*)
           (*PRINT-LENGTH* XCL:*TRACE-LENGTH*))
          ,@(CASE LAMBDA-CAR
                  ((IL:LAMBDA IL:NLAMBDA)
                     (IL:IF (LISTP ARG-LIST)
                        IL:THEN
                              ;; Interlisp spread function.  The ARG-LIST is, in fact, a list of argument names.
                              `((LET (($$INDENT$$ (+ 10 (* XCL:*TRACE-DEPTH* 4))))
                                     ,@(IL:FOR VAR IL:IN ARG-LIST IL:COLLECT `(PRINT-TRACED-ARGUMENT
                                                                               ',VAR
                                                                               ,VAR $$INDENT$$))))
                        IL:ELSEIF (EQ LAMBDA-CAR 'IL:LAMBDA)
                          IL:THEN
                              ;; Interlisp Lambda no-spread function.  Print out at most *TRACE-LENGTH* arguments.
                              `((IL:BIND ($$INDENT$$ IL:_ (+ 10 (* XCL:*TRACE-DEPTH* 4))) IL:FOR $ARG-COUNTER$
                                   IL:FROM 1 IL:TO (IF (NULL XCL:*TRACE-LENGTH*)
                                                       ,ARG-LIST
                                                       (MIN XCL:*TRACE-LENGTH* ,ARG-LIST))
                                   IL:DO (PRINT-TRACED-ARGUMENT $ARG-COUNTER$ (IL:ARG ,ARG-LIST $ARG-COUNTER$)
                                                                $$INDENT$$)))
                        IL:ELSE
                              ;; Interlisp NLambda no-spread function.  Print out at most *TRACE-LENGTH* arguments.  Also, be careful to check
                              ;; that the argument list is really a list.
                              `((LET (($$INDENT$$ (+ 10 (* XCL:*TRACE-DEPTH* 4))))
                                     (IF (LISTP ,ARG-LIST)
                                         (IL:FOR $ARGUMENT$ IL:IN ,ARG-LIST IL:AS $ARG-COUNTER$ IL:FROM 1
                                            IL:WHILE (OR (NULL XCL:*TRACE-LENGTH*)
                                                         (<= $ARG-COUNTER$ XCL:*TRACE-LENGTH*))
                                            IL:DO (PRINT-TRACED-ARGUMENT $ARG-COUNTER$ $ARGUMENT$ $$INDENT$$))
                                         (PRINT-TRACED-ARGUMENT ',ARG-LIST ,ARG-LIST $$INDENT$$))))))
                  ((LAMBDA)
                     ;; A Common Lisp function.
                     (MULTIPLE-VALUE-BIND (REQUIRED OPTIONAL REST KEY KEY-APPEARED? ALLOW-OTHER-KEYS)
                         (PARSE-CL-ARGLIST ARG-LIST)
                       `((PRINT-TRACED-CL-ARGLIST XCL:ARGLIST ',REQUIRED ',OPTIONAL ',REST ',KEY ,KEY-APPEARED?
                                ,ALLOW-OTHER-KEYS
                                (+ 8 (* XCL:*TRACE-DEPTH* 4))
                                XCL:*TRACE-VERBOSE*))))))))

(DEFUN PRINT-TRACE-ENTRY-INFO (TRACED-FN IN-FN)
   (DECLARE (SPECIAL IL:BOLDFONT IL:DEFAULTFONT))
   (IL:SPACES (* XCL:*TRACE-DEPTH* 4))
   (PRINC (1+ XCL:*TRACE-DEPTH*))
   (PRINC " - Enter ")
   (IL:CHANGEFONT IL:BOLDFONT)
   (PRIN1 TRACED-FN)
   (IL:CHANGEFONT IL:DEFAULTFONT)
   (WHEN (NOT (NULL IN-FN))
       (PRINC " in ")
       (IL:CHANGEFONT IL:BOLDFONT)
       (PRIN1 IN-FN)
       (IL:CHANGEFONT IL:DEFAULTFONT))
   (PRINC ":")
   (TERPRI))
```

```
(DEFUN PRINT-TRACE-EXIT-INFO (TRACED-FN IN-FN FN-VALUES)
   (DECLARE (SPECIAL IL:BOLDFONT IL:DEFAULTFONT))
   (IL:SPACES (* XCL:*TRACE-DEPTH* 4))
   (PRINC (1+ XCL:*TRACE-DEPTH*))
   (PRINC " - Exit ")
   (IL:CHANGEFONT IL:BOLDFONT)
   (PRIN1 TRACED-FN)
   (IL:CHANGEFONT IL:DEFAULTFONT)
   (WHEN (NOT (NULL IN-FN))
       (PRINC " in ")
       (IL:CHANGEFONT IL:BOLDFONT)
       (PRIN1 IN-FN)
       (IL:CHANGEFONT IL:DEFAULTFONT))
   (PRINC " =>")
   (TERPRI)
   (IL:FOR VALUE IL:IN FN-VALUES IL:DO (IL:SPACES (+ 10 (* XCL:*TRACE-DEPTH* 4)))
                                       (PRIN1 VALUE)
                                       (TERPRI)))

(DEFUN PRINT-TRACED-ARGUMENT (NAME VALUE INDENT &OPTIONAL PRIN1-THE-NAME?)
   (IL:SPACES INDENT)
   (WHEN (TYPEP NAME 'FIXNUM)
         (PRINC "Arg "))
   (IF PRIN1-THE-NAME?
       (PRIN1 NAME)
       (PRINC NAME))
   (PRINC " = ")
   (PRIN1 VALUE)
   (TERPRI))

(DEFUN PRINT-TRACED-CL-ARGLIST (ARGS REQUIRED OPTIONAL REST KEY KEY-APPEARED? ALLOW-OTHER-KEYS
                                     SMALL-INDENT VERBOSE?)
   (DECLARE (SPECIAL IL:BOLDFONT IL:DEFAULTFONT))
   (LET* ((INDENT (+ SMALL-INDENT 2)))
         (WHEN REQUIRED
             (IL:FOR VAR IL:IN REQUIRED IL:DO (COND
                                                ((NULL ARGS)
                                                 (IL:SPACES INDENT)
                                                 (PRINC VAR)
                                                 (IL:CHANGEFONT IL:BOLDFONT)
                                                 (PRINC " ** NOT SUPPLIED **")
                                                 (IL:CHANGEFONT IL:DEFAULTFONT)
                                                 (TERPRI))
                                                (T (PRINT-TRACED-ARGUMENT VAR (POP ARGS)
                                                          INDENT)))))
         (WHEN OPTIONAL
             (WHEN VERBOSE?
                 (IL:SPACES SMALL-INDENT)
                 (PRINC '&OPTIONAL)
                 (TERPRI))
             (IL:FOR VAR IL:IN OPTIONAL IL:DO (IF (NULL ARGS)
                                                  (WHEN VERBOSE?
                                                      (IL:SPACES INDENT)
                                                      (PRINC VAR)
                                                      (PRINC " not supplied")
                                                      (TERPRI))
                                                  (PRINT-TRACED-ARGUMENT VAR (POP ARGS)
                                                        INDENT))))
         (WHEN REST
             (WHEN VERBOSE?
                 (IL:SPACES SMALL-INDENT)
                 (PRINC '&REST)
                 (TERPRI))
             (PRINT-TRACED-ARGUMENT REST ARGS INDENT))
         (WHEN KEY
             (WHEN VERBOSE?
                 (IL:SPACES SMALL-INDENT)
                 (PRINC '&KEY)
                 (TERPRI))
             (IL:FOR VAR IL:IN KEY IL:DO (IL:FOR TAIL IL:ON ARGS IL:BY CDDR IL:DO (WHEN (EQ VAR (CAR TAIL))
                                                                                     (PRINT-TRACED-ARGUMENT
                                                                                      VAR
                                                                                      (CADR TAIL)
                                                                                      INDENT T)
                                                                                     (RETURN)))))
         (WHEN KEY-APPEARED?
             (LET (TEMP)
                  (COND
                     ((ODDP (LENGTH ARGS))
                      (IL:SPACES SMALL-INDENT)
                      (IL:CHANGEFONT IL:BOLDFONT)
                      (PRINC "** Odd-length &KEY argument list: **")
                      (IL:CHANGEFONT IL:DEFAULTFONT)
```

```
                    (TERPRI)
                    (IL:SPACES INDENT)
                    (PRIN1 ARGS)
                    (TERPRI))
                   ((SETQ TEMP (IL:FIND KEYWORD IL:IN ARGS IL:BY (CDDR KEYWORD)
                                      IL:SUCHTHAT (IF ALLOW-OTHER-KEYS
                                                      (NOT (KEYWORDP KEYWORD))
                                                      (NOT (MEMBER KEYWORD KEY :TEST 'EQ)))))
                    (IL:SPACES SMALL-INDENT)
                    (IL:CHANGEFONT IL:BOLDFONT)
                    (PRINC "** Illegal &KEY argument: **")
                    (IL:CHANGEFONT IL:DEFAULTFONT)
                    (TERPRI)
                    (IL:SPACES INDENT)
                    (PRIN1 TEMP)
                    (TERPRI)))))
          (WHEN (AND (NOT REST)
                     (NOT KEY-APPEARED?)
                     (NOT (NULL ARGS)))
              (IL:SPACES SMALL-INDENT)
              (IL:CHANGEFONT IL:BOLDFONT)
              (PRINC "** Extra arguments: **")
              (IL:CHANGEFONT IL:DEFAULTFONT)
              (TERPRI)
              (IL:SPACES INDENT)
              (PRIN1 ARGS)
              (TERPRI))))
```

(DEFVAR **XCL:*TRACE-LEVEL*** NIL

;;; What to bind *PRINT-LEVEL* to when printing argument values in TRACE output.

)

(DEFVAR **XCL:*TRACE-LENGTH*** NIL

;;; What to bind *PRINT-LENGTH* to during the printing of argument values in TRACE output. Also controls the number of arguments to no-spread
;;; functions that will be printed.

)

(DEFVAR **XCL:*TRACE-VERBOSE*** T

;;; Controls whether or not various parts of TRACE output are printed:

;; The lambda-list keywords &OPTIONAL, &REST, and &KEY.

;; Trailing unsupplied &OPTIONAL arguments.

)

(DEFVAR ***TRACE-OUTPUT*** (**XCL:CREATE-TRACE-WINDOW**))

(IL:DEFINEQ

(**TRACE**

```
  (IL:NLAMBDA CL::FNS                                                    ; Edited 2-Apr-87 16:10 by Pavel
    (SETQ CL::FNS (IL:NLAMBDA.ARGS CL::FNS))
    (IF (NULL CL::FNS)
        XCL::*TRACED-FNS*
        (IL:FOR CL::FN IL:IN CL::FNS IL:JOIN (IF (CONSP CL::FN)
                                                 (XCL:TRACE-FUNCTION (FIRST CL::FN)
                                                        :IN
                                                        (THIRD CL::FN))
                                                 (XCL:TRACE-FUNCTION CL::FN))))))
```

(**UNTRACE**

```
  (IL:NLAMBDA CL::FNS                                                    ; Edited 2-Apr-87 16:39 by Pavel
    (SETQ CL::FNS (IL:NLAMBDA.ARGS CL::FNS))
    (FLET ((CL::UNTRACE-ENTRY (CL::ENTRY)
                (IF (CONSP CL::ENTRY)
                    (XCL:UNBREAK-FUNCTION (FIRST CL::ENTRY)
                           :IN
                           (SECOND CL::ENTRY))
                    (XCL:UNBREAK-FUNCTION CL::ENTRY))))
          (COND
             ((NULL CL::FNS)
              (IL:FOR CL::ENTRY IL:IN (REVERSE XCL::*TRACED-FNS*) IL:JOIN (CL::UNTRACE-ENTRY CL::ENTRY)))
             ((EQUAL CL::FNS '(T))
              (WHEN XCL::*TRACED-FNS*
                  (CL::UNTRACE-ENTRY (CAR XCL::*TRACED-FNS*))))
             (T (IL:FOR CL::FN IL:IN CL::FNS IL:JOIN (IF (CONSP CL::FN)
                                                         (XCL:UNBREAK-FUNCTION (FIRST CL::FN)
```

```
                                                                    :IN
                                                                    (THIRD CL::FN))
                                                            (XCL:UNBREAK-FUNCTION CL::FN))))))))

)


(DEFUN XCL:TRACE-FUNCTION (XCL::FN-TO-TRACE &KEY ((:IN XCL::IN-FN))
                                    XCL::REBREAK?)
   (COND
      ((CONSP XCL::FN-TO-TRACE)
       (IL:FOR XCL::FN IL:IN XCL::FN-TO-TRACE IL:JOIN (XCL:TRACE-FUNCTION XCL::FN :IN XCL::IN-FN)))
      ((CONSP XCL::IN-FN)
       (IL:FOR XCL::FN IL:IN XCL::IN-FN IL:JOIN (XCL:TRACE-FUNCTION XCL::FN-TO-TRACE :IN XCL::FN)))
      ((NULL (IL:GETD XCL::FN-TO-TRACE))
       (ERROR 'XCL:UNDEFINED-FUNCTION :NAME XCL::FN-TO-TRACE)
       NIL)
      ((IL:UNSAFE.TO.MODIFY XCL::FN-TO-TRACE "trace")
       (FORMAT *ERROR-OUTPUT* "~S not traced.~%" XCL::FN-TO-TRACE)
       NIL)
      (T (XCL:UNBREAK-FUNCTION XCL::FN-TO-TRACE :IN XCL::IN-FN :NO-ERROR T)
         (UNLESS XCL::REBREAK?                                    ; Save the breaking information for REBREAK, but don't save it if
                                                                  ; we're being called from REBREAK itself.
             (SETF (GETHASH (IF (NULL XCL::IN-FN)
                                XCL::FN-TO-TRACE
                                `(,XCL::FN-TO-TRACE :IN ,XCL::IN-FN))
                            XCL::*BREAK-HASH-TABLE*)
                   (LIST XCL::FN-TO-TRACE :IN XCL::IN-FN :TRACE? T :REBREAK? T)))
         (IF (NULL XCL::IN-FN)
             (LET ((XCL::ORIGINAL (LET ((*PRINT-CASE* :UPCASE))
                                    (MAKE-SYMBOL (FORMAT NIL "Original ~A" XCL::FN-TO-TRACE)))))
                  (IL:PUTD XCL::ORIGINAL (IL:GETD XCL::FN-TO-TRACE)
                           T)
                  (IL:PUTD XCL::FN-TO-TRACE (COMPILE NIL (CREATE-TRACED-DEFINITION XCL::FN-TO-TRACE NIL
                                                                                   XCL::ORIGINAL))
                           T)
                  (SETF (GET XCL::FN-TO-TRACE 'IL:BROKEN)
                        XCL::ORIGINAL)
                  (PUSH XCL::FN-TO-TRACE IL:BROKENFNS)
                  (PUSH XCL::FN-TO-TRACE XCL::*TRACED-FNS*)
                  (SETQ XCL::*UNBROKEN-FNS* (DELETE XCL::FN-TO-TRACE XCL::*UNBROKEN-FNS*))
                  (LIST XCL::FN-TO-TRACE))
             (LET ((XCL::MIDDLE-MAN (CONSTRUCT-MIDDLE-MAN XCL::FN-TO-TRACE XCL::IN-FN)))
                  (IF (NOT (HAS-CALLS XCL::IN-FN XCL::FN-TO-TRACE))
                      (ERROR "~S is not called from ~S." XCL::FN-TO-TRACE XCL::IN-FN))
                  (COMPILE XCL::MIDDLE-MAN (CREATE-TRACED-DEFINITION XCL::FN-TO-TRACE XCL::IN-FN
                                                                     XCL::FN-TO-TRACE))
                  (CHANGE-CALLS XCL::FN-TO-TRACE XCL::MIDDLE-MAN XCL::IN-FN 'UNBREAK-FROM-RESTORE-CALLS)
                  (LET ((XCL::ENTRY (LIST XCL::FN-TO-TRACE XCL::IN-FN XCL::MIDDLE-MAN)))
                       (PUSH XCL::ENTRY IL:BROKENFNS)
                       (PUSH XCL::ENTRY XCL::*TRACED-FNS*))
                  (SETQ XCL::*UNBROKEN-FNS* (DELETE `(,XCL::FN-TO-TRACE :IN ,XCL::IN-FN)
                                                    XCL::*UNBROKEN-FNS* :TEST 'EQUAL))
                  (LIST `(,XCL::FN-TO-TRACE :IN ,XCL::IN-FN))))))))
```

;;; Support for breaking.

```
(DEFUN XCL:BREAK-FUNCTION (XCL::FN-TO-BREAK &KEY ((:IN XCL::IN-FN))
                                    ((:WHEN XCL::WHEN-EXPR)
                                     T)
                                    XCL::TRACE? XCL::REBREAK?)
   (COND
      (XCL::TRACE? (XCL:TRACE-FUNCTION XCL::FN-TO-BREAK :IN XCL::IN-FN :REBREAK? XCL::REBREAK?))
      ((CONSP XCL::FN-TO-BREAK)
       (IL:FOR XCL::FN IL:IN XCL::FN-TO-BREAK IL:JOIN (XCL:BREAK-FUNCTION XCL::FN :IN XCL::IN-FN :WHEN
                                                                XCL::WHEN-EXPR :REBREAK? XCL::REBREAK?)))
      ((CONSP XCL::IN-FN)
       (IL:FOR XCL::FN IL:IN XCL::IN-FN IL:JOIN (XCL:BREAK-FUNCTION XCL::FN-TO-BREAK :IN XCL::FN :WHEN
                                                          XCL::WHEN-EXPR :REBREAK? XCL::REBREAK?)))
      ((IL:UNSAFE.TO.MODIFY XCL::FN-TO-BREAK "break")
       (FORMAT *ERROR-OUTPUT* "~S not broken." XCL::FN-TO-BREAK)
       NIL)
      (T (UNLESS XCL::REBREAK?                                    ; Save the breaking information for REBREAK.  Don't do it,
                                                                  ; though, if we're being called from REBREAK.
             (SETF (GETHASH (IF (NULL XCL::IN-FN)
                                XCL::FN-TO-BREAK
                                `(,XCL::FN-TO-BREAK :IN ,XCL::IN-FN))
                            XCL::*BREAK-HASH-TABLE*)
                   (LIST XCL::FN-TO-BREAK :IN XCL::IN-FN :WHEN XCL::WHEN-EXPR :REBREAK? T)))
         (WHEN (EQ XCL::WHEN-EXPR :ONCE)
             (SETQ XCL::WHEN-EXPR `(FUNCALL ',(LET ((XCL::TRIGGERED-YET? NIL))
                                                #'(LAMBDA NIL (IF XCL::TRIGGERED-YET?
                                                                  NIL
                                                                  (SETQ XCL::TRIGGERED-YET? T)))))))
         (XCL:UNBREAK-FUNCTION XCL::FN-TO-BREAK :IN XCL::IN-FN :NO-ERROR T)
```

```
         (IF (NULL XCL::IN-FN)
             (LET* ((XCL::ORIGINAL-DEF (OR (IL:GETD XCL::FN-TO-BREAK)
                                           (ERROR 'XCL:UNDEFINED-FUNCTION :NAME XCL::FN-TO-BREAK)))
                    (XCL::ORIGINAL (LET ((*PRINT-CASE* :UPCASE))
                                     (MAKE-SYMBOL (FORMAT NIL "Original ~A" XCL::FN-TO-BREAK)))))
                   (IL:PUTD XCL::ORIGINAL XCL::ORIGINAL-DEF T)
                   (IL:PUTD XCL::FN-TO-BREAK (COMPILE NIL (CREATE-BROKEN-DEFINITION XCL::FN-TO-BREAK
                                                                                    XCL::FN-TO-BREAK XCL::ORIGINAL XCL::WHEN-EXPR
                                                                                    XCL::FN-TO-BREAK))
                          T)
                   (SETF (GET XCL::FN-TO-BREAK 'IL:BROKEN)
                         XCL::ORIGINAL)
                   (PUSH XCL::FN-TO-BREAK IL:BROKENFNS)
                   (SETQ XCL::*UNBROKEN-FNS* (DELETE XCL::FN-TO-BREAK XCL::*UNBROKEN-FNS*))
                   (LIST XCL::FN-TO-BREAK))
             (LET ((XCL::MIDDLE-MAN (CONSTRUCT-MIDDLE-MAN XCL::FN-TO-BREAK XCL::IN-FN)))
                  (IF (NOT (HAS-CALLS XCL::IN-FN XCL::FN-TO-BREAK))
                      (ERROR "~S is not called from ~S." XCL::FN-TO-BREAK XCL::IN-FN))
                  (XCL:UNADVISE-FUNCTION XCL::FN-TO-BREAK :IN XCL::IN-FN :NO-ERROR T)
                  (COMPILE XCL::MIDDLE-MAN (CREATE-BROKEN-DEFINITION XCL::FN-TO-BREAK XCL::MIDDLE-MAN
                                                                     XCL::FN-TO-BREAK XCL::WHEN-EXPR `(,XCL::FN-TO-BREAK
                                                                                                       :IN
                                                                                                       ,XCL::IN-FN)))
                  (CHANGE-CALLS XCL::FN-TO-BREAK XCL::MIDDLE-MAN XCL::IN-FN 'UNBREAK-FROM-RESTORE-CALLS)
                  (PUSH (LIST XCL::FN-TO-BREAK XCL::IN-FN XCL::MIDDLE-MAN)
                        IL:BROKENFNS)
                  (SETQ XCL::*UNBROKEN-FNS* (DELETE `(,XCL::FN-TO-BREAK :IN ,XCL::IN-FN)
                                                    XCL::*UNBROKEN-FNS* :TEST 'EQUAL))
                  (LIST `(,XCL::FN-TO-BREAK :IN ,XCL::IN-FN))))))

(DEFUN XCL:UNBREAK-FUNCTION (XCL::BROKEN-FN &KEY ((:IN XCL::IN-FN))
                                                 XCL::NO-ERROR)
   (COND
      ((CONSP XCL::BROKEN-FN)
       (IL:FOR XCL::FN IL:IN XCL::BROKEN-FN IL:JOIN (XCL:UNBREAK-FUNCTION XCL::FN :IN XCL::IN-FN)))
      ((CONSP XCL::IN-FN)
       (IL:FOR XCL::FN IL:IN XCL::IN-FN IL:JOIN (XCL:UNBREAK-FUNCTION XCL::BROKEN-FN :IN XCL::FN)))
      ((NULL XCL::IN-FN)
       (LET ((XCL::ORIGINAL (GET XCL::BROKEN-FN 'IL:BROKEN)))
            (COND
               ((NULL XCL::ORIGINAL)
                (UNLESS XCL::NO-ERROR (FORMAT *ERROR-OUTPUT* "~S is not broken.~%" XCL::BROKEN-FN))
                NIL)
               (T (IL:PUTD XCL::BROKEN-FN (IL:GETD XCL::ORIGINAL)
                           T)
                  (REMPROP XCL::BROKEN-FN 'IL:BROKEN)
                  (SETQ IL:BROKENFNS (DELETE XCL::BROKEN-FN IL:BROKENFNS))
                  (SETQ XCL::*TRACED-FNS* (DELETE XCL::BROKEN-FN XCL::*TRACED-FNS*))
                  (PUSH XCL::BROKEN-FN XCL::*UNBROKEN-FNS*)
                  (LIST XCL::BROKEN-FN)))))
      (T (LET* ((XCL::ENTRY (FIND-IF #'(LAMBDA (XCL::ENTRY)
                                         (AND (CONSP XCL::ENTRY)
                                              (EQ (FIRST XCL::ENTRY)
                                                  XCL::BROKEN-FN)
                                              (EQ (SECOND XCL::ENTRY)
                                                  XCL::IN-FN)))
                                     IL:BROKENFNS))
                (XCL::MIDDLE-MAN (THIRD XCL::ENTRY)))
               (COND
                  ((NULL XCL::ENTRY)
                   (UNLESS XCL::NO-ERROR (FORMAT *ERROR-OUTPUT* "~S :IN ~S is not broken.~%" XCL::BROKEN-FN
                                                 XCL::IN-FN))
                   NIL)
                  (T (CHANGE-CALLS XCL::MIDDLE-MAN XCL::BROKEN-FN XCL::IN-FN)
                     (FINISH-UNBREAKING XCL::BROKEN-FN XCL::IN-FN XCL::MIDDLE-MAN XCL::ENTRY)
                     (LIST `(,XCL::BROKEN-FN :IN ,XCL::IN-FN)))))))))

(DEFUN XCL:REBREAK-FUNCTION (XCL::FN-TO-REBREAK &KEY ((:IN XCL::IN-FN)))
   (COND
      ((CONSP XCL::FN-TO-REBREAK)
       (IL:FOR XCL::FN IL:IN XCL::FN-TO-REBREAK IL:JOIN (XCL:REBREAK-FUNCTION XCL::FN :IN XCL::IN-FN)))
      ((CONSP XCL::IN-FN)
       (IL:FOR XCL::FN IL:IN XCL::IN-FN IL:JOIN (XCL:REBREAK-FUNCTION XCL::FN-TO-REBREAK :IN XCL::FN)))
      (T (LET* ((XCL::NAME (IF (NULL XCL::IN-FN)
                               XCL::FN-TO-REBREAK
                               `(,XCL::FN-TO-REBREAK :IN ,XCL::IN-FN)))
                (XCL::INFO (GETHASH XCL::NAME XCL::*BREAK-HASH-TABLE*)))
               (COND
                  ((NULL XCL::INFO)
                   (FORMAT *ERROR-OUTPUT* "~S has never been broken.~%" XCL::NAME)
                   NIL)
                  (T (APPLY 'XCL:BREAK-FUNCTION XCL::INFO)))))))

(DEFUN CREATE-BROKEN-DEFINITION (WRAPPED-FN-NAME BROKEN-FN-NAME FN-TO-CALL WHEN-EXPR BREAKPOINT-NAME)
```

;;; WRAPPED-FN-NAME must be the symbol naming the function that will break when it is called.

;;; BROKEN-FN-NAME is the symbol in whose function cell our lambda-form will be put.

;;; FN-TO-CALL is the function-object to be FUNCALL'ed when we want to call the unbroken version of the wrapped function.

;;; BREAKPOINT-NAME is the value the debugger will use for BRKFN.

;;; We return a lambda-form suitable for being called in order to (possibly) activate the breakpoint.

```
   (MULTIPLE-VALUE-BIND (LAMBDA-CAR ARG-LIST CALLING-FORM)
       (FUNCTION-WRAPPER-INFO WRAPPED-FN-NAME FN-TO-CALL)
     `(,LAMBDA-CAR ,(IF (EQ LAMBDA-CAR 'LAMBDA)
                        '(&REST XCL:ARGLIST)
                        ARG-LIST)
             ,@(AND ARG-LIST (MEMBER LAMBDA-CAR '(IL:LAMBDA IL:NLAMBDA))
                    `((DECLARE (SPECIAL ,@(IF (SYMBOLP ARG-LIST)
                                              (LIST ARG-LIST)
                                              ARG-LIST)))))
             (IL:\\CALLME '(:BROKEN ,BREAKPOINT-NAME))
             (IF ,WHEN-EXPR
                 (LET (($POS$ (IL:STKNTH -1)))
                      (UNWIND-PROTECT
                          (XCL:DEBUGGER :FORM `(FUNCALL ',#'(LAMBDA NIL ,CALLING-FORM))
                                 :ENVIRONMENT NIL :STACK-POSITION $POS$ :CONDITION
                                 ',(XCL:MAKE-CONDITION 'BREAKPOINT :FUNCTION BREAKPOINT-NAME))
                          (IL:RELSTK $POS$)))
                 ,CALLING-FORM))))
```

(DEFUN **UNBREAK-FROM-RESTORE-CALLS** (FROM TO FN)

;;; Somebody has restored all of the changed calls in FN, including one we made, changing calls to FROM into calls to TO. This came about from ;;; breaking (FROM :IN FN), where TO was the middle-man. Undo that breaking.

```
   (LET ((ENTRY (FIND-IF #'(LAMBDA (ENTRY)
                                (AND (CONSP ENTRY)
                                     (EQ (FIRST ENTRY)
                                         FROM)
                                     (EQ (SECOND ENTRY)
                                         FN)))
                         IL:BROKENFNS)))
        (ASSERT (EQ TO (THIRD ENTRY))
               NIL "BUG: Inconsistency in SI::UNBREAK-FROM-RESTORE-CALLS")
        (FINISH-UNBREAKING FROM FN TO ENTRY)
        (FORMAT *TERMINAL-IO* "(~S :IN ~S) unbroken.~%" FROM FN)))
```

(DEFUN **FINISH-UNBREAKING** (BROKEN-FN IN-FN MIDDLE-MAN ENTRY)

```
   (SETQ IL:BROKENFNS (DELETE ENTRY IL:BROKENFNS))
   (SETQ XCL::*TRACED-FNS* (DELETE ENTRY XCL::*TRACED-FNS*))
   (PUSH `(,BROKEN-FN :IN ,IN-FN)
         XCL::*UNBROKEN-FNS*))
```

(DEFVAR **IL:BROKENFNS** NIL)

(DEFVAR **XCL::*BREAK-HASH-TABLE*** (MAKE-HASH-TABLE :TEST 'EQUAL))

(DEFVAR **XCL::*UNBROKEN-FNS*** NIL)

(IL:PUTPROPS **IL:BROKEN** **IL:PROPTYPE** IGNORE)

;; The old Interlisp interface to breaking.

(IL:DEFINEQ

(IL:BREAK

```
  (IL:NLAMBDA IL:X                                                    ; Edited 13-Apr-87 13:51 by Pavel
    (IL:FOR IL:X IL:IN (IL:NLAMBDA.ARGS IL:X) IL:JOIN (IL:IF (OR (IL:LITATOM IL:X)
                                                                  (IL:STRING.EQUAL (CADR IL:X)
                                                                                   "IN"))
                                                              IL:THEN (IL:BREAK0 IL:X T)
                                                              IL:ELSE (IL:APPLY 'IL:BREAK0 IL:X)))))
```

(IL:BREAK0

```
  (IL:LAMBDA (IL:FN IL:WHEN IL:COMS IL:BRKFN)                         ; Edited 18-Apr-87 18:56 by Pavel
    (WHEN IL:COMS (CERROR "Ignore COMS" "Break 'commands' ~S no longer supported." IL:COMS))
    (WHEN (AND IL:BRKFN (IL:NEQ IL:BRKFN 'IL:BREAK1))
          (CERROR "Ignore BRKFN" "Unexpected BRKFN passed to BREAK0: ~S" IL:BRKFN))
    (WHEN (NULL IL:WHEN)
          (IL:SETQ IL:WHEN T))
    (COND
```

```
       ((IL:LISTP IL:FN)
        (COND
           ((IL:STRING.EQUAL (SECOND IL:FN)
                  "IN")
            (XCL:BREAK-FUNCTION (FIRST IL:FN)
                   :IN
                   (THIRD IL:FN)
                   :WHEN IL:WHEN))
           (T (IL:FOR IL:X IL:IN IL:FN IL:JOIN (IL:BREAK0 IL:X IL:WHEN)))))
       (T (XCL:BREAK-FUNCTION IL:FN :WHEN IL:WHEN)))))
```

(IL:REBREAK

```
  (IL:NLAMBDA IL:FNS                                                    ; Edited 3-Apr-87 12:07 by Pavel
    (IL:SETQ IL:FNS (IL:NLAMBDA.ARGS IL:FNS))
    (FLET ((IL:REBREAK-FN (IL:FN)
                  (IL:IF (IL:LISTP IL:FN)
                      IL:THEN (XCL:REBREAK-FUNCTION (FIRST IL:FN)
                                     :IN
                                     (THIRD IL:FN))
                       IL:ELSE (XCL:REBREAK-FUNCTION IL:FN))))
          (COND
             ((NULL IL:FNS)
              (IL:FOR IL:FN IL:IN XCL::*UNBROKEN-FNS* IL:JOIN (IL:REBREAK-FN IL:FN)))
             ((IL:EQUAL IL:FNS '(T))
              (AND (NOT (NULL XCL::*UNBROKEN-FNS*))
                   (IL:REBREAK-FN (CAR XCL::*UNBROKEN-FNS*))))
             (T (IL:FOR IL:FN IL:IN IL:FNS IL:JOIN (IL:REBREAK-FN IL:FN)))))))
```

(XCL:UNBREAK

```
  (IL:NLAMBDA XCL::FNS                                                  ; Edited 2-Apr-87 16:39 by Pavel
    (SETQ XCL::FNS (IL:NLAMBDA.ARGS XCL::FNS))
    (FLET ((XCL::UNBREAK-ENTRY (XCL::ENTRY)
                  (IF (CONSP XCL::ENTRY)
                      (XCL:UNBREAK-FUNCTION (FIRST XCL::ENTRY)
                             :IN
                             (SECOND XCL::ENTRY))
                      (XCL:UNBREAK-FUNCTION XCL::ENTRY))))
          (COND
             ((NULL XCL::FNS)
              (IL:FOR XCL::ENTRY IL:IN (REVERSE IL:BROKENFNS) IL:JOIN (XCL::UNBREAK-ENTRY XCL::ENTRY)))
             ((EQUAL XCL::FNS '(T))
              (WHEN IL:BROKENFNS
                   (XCL::UNBREAK-ENTRY (CAR IL:BROKENFNS))))
             (T (IL:FOR XCL::FN IL:IN XCL::FNS IL:JOIN (IF (CONSP XCL::FN)
                                                           (XCL:UNBREAK-FUNCTION (FIRST XCL::FN)
                                                                  :IN
                                                                  (THIRD XCL::FN))
                                                           (XCL:UNBREAK-FUNCTION XCL::FN))))))))
```

(IL:UNBREAK0

```
  (IL:LAMBDA (IL:FN)                                                    ; Edited 1-Apr-87 22:12 by Pavel
    (IL:IF (IL:LISTP IL:FN)
        IL:THEN (XCL:UNBREAK-FUNCTION (CAR IL:FN)
                        :IN
                        (CADDR IL:FN))
      IL:ELSE (XCL:UNBREAK-FUNCTION IL:FN))))

)

(IL:DEFINEQ
```

(IL:BREAK1

```
  (IL:NLAMBDA (IL:BRKEXP IL:BRKWHEN IL:BRKFN IL:BRKCOMS IL:BRKTYPE XCL:CONDITION)
                                                                        ; Edited 24-Mar-87 16:07 by amd
    (IL:|if| (EVAL IL:BRKWHEN)
        IL:|then|
              ;; should probably default CONDITION depending on BRKTYPE to interrupt, breakpoint error, etc.
              (WHEN IL:BRKCOMS (IL:PRINTOUT T "BRKCOMS no longer supported:" IL:BRKCOMS T))
              (LET ((IL:POS (IL:STKNTH 0 IL:BRKFN)))
                   (UNWIND-PROTECT
                       (XCL:DEBUGGER :FORM IL:BRKEXP :ENVIRONMENT NIL :STACK-POSITION IL:POS :CONDITION
                              (OR XCL:CONDITION (XCL:MAKE-CONDITION 'BREAKPOINT :FUNCTION IL:BRKFN)))
                       (IL:RELSTK IL:POS)))
      IL:|else| (EVAL IL:BRKEXP))))

)


(XCL:DEFINE-SPECIAL-FORM IL:BREAK1 (&OPTIONAL IL:EXP IL:WHEN IL:FN IL:COMS TYPE XCL:CONDITION &ENVIRONMENT
                                              IL:ENV)
   (IL:IF (EVAL IL:WHEN IL:ENV)
       IL:THEN (WHEN IL:COMS (IL:PRINTOUT T "BRKCOMS no longer supported:" IL:COMS T))
              (LET ((IL:POS (IL:STKNTH 0 IL:FN)))
```

```
                (UNWIND-PROTECT
                    (XCL:DEBUGGER :FORM IL:EXP :ENVIRONMENT IL:ENV :STACK-POSITION IL:POS :CONDITION
                           (OR XCL:CONDITION (XCL:MAKE-CONDITION 'BREAKPOINT :FUNCTION IL:FN)))
                    (IL:RELSTK IL:POS)))
     IL:ELSE (EVAL IL:EXP IL:ENV)))


(XCL:DEFOPTIMIZER IL:BREAK1 (&OPTIONAL IL:EXP IL:WHEN IL:FN IL:COMS TYPE XCL:CONDITION)
                            (WHEN IL:COMS (IL:PRINTOUT T "BRKCOMS no longer supported:" IL:COMS T))
                            `(FLET
                              (($BRKEXP$ NIL ,IL:EXP))
                              (IL:IF ,IL:WHEN
                                  IL:THEN
                                  (LET (($POS$ (IL:STKNTH 0 ',IL:FN)))
                                       (UNWIND-PROTECT
                                           (XCL:DEBUGGER
                                            :FORM
                                            `(FUNCALL ',#'$BRKEXP$)
                                            :ENVIRONMENT NIL :STACK-POSITION $POS$ :CONDITION
                                            ,(OR XCL:CONDITION `(IL:LOADTIMECONSTANT
                                                                 (XCL:MAKE-CONDITION 'BREAKPOINT :FUNCTION
                                                                        ',IL:FN))))
                                           (IL:RELSTK $POS$)))
                                IL:ELSE ($BRKEXP$))))
```

;; Arrange for the proper compiler and package

```
(IL:PUTPROPS IL:BREAK-AND-TRACE IL:FILETYPE :COMPILE-FILE)

(IL:PUTPROPS IL:BREAK-AND-TRACE IL:MAKEFILE-ENVIRONMENT (:READTABLE "XCL" :PACKAGE "SYSTEM"))

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

(IL:ADDTOVAR IL:NLAMA XCL:UNBREAK IL:REBREAK IL:BREAK UNTRACE TRACE)

(IL:ADDTOVAR IL:NLAML IL:BREAK1)

(IL:ADDTOVAR IL:LAMA )
)

(IL:PUTPROPS IL:BREAK-AND-TRACE IL:COPYRIGHT ("Venue & Xerox Corporation" 1987 1988 1990))
```

FUNCTION INDEX

VARIABLE INDEX

PROPERTY INDEX

OPTIMIZER INDEX

SPECIAL-FORM INDEX