

File created: 16-May-90 11:55:52 {DSK}<usr>local>lde>lispcore>sources>ADVISE.;

changes to: (IL:VARS IL:ADVISECOMS)

previous date: 15-Aug-88 12:29:50 {DSK}<usr>local>lde>lispcore>sources>ADVISE.;

Read Table: XCL

Package: SYSTEM

Format: XCCS

; Copyright (c) 1978, 1984, 1986, 1987, 1988, 1990 by Venue & Xerox Corporation. All rights reserved.
; The following program was created in 1978 but has not been published
; within the meaning of the copyright law, is furnished under license,
; and may not be used, copied and/or disclosed except in accordance
; with the terms of said license.

```
(IL:RPAQQ IL:ADVISECOMS
  ((IL:STRUCTURES ADVICE)
   (IL:VARIABLES IL:ADVISED-FNS *UNADVISED-FNS*)
   ;;
   ;; Interlisp entry points.
   (IL:FNS IL:ADVISE IL:UNADVISE IL:READWISE)
   (IL:PROP IL:ARGNAMES IL:ADVISE)
   ;;
   ;; XCL entry points.
   (IL:FUNCTIONS XCL:ADVISE-FUNCTION XCL:UNADVISE-FUNCTION XCL:READWISE-FUNCTION)
   (IL:FUNCTIONS UNADVISE-FROM-RESTORE-CALLS FINISH-ADVISING FINISH-UNADVISING)
   ;;
   ;; The advice database.
   (IL:VARIABLES *ADVICE-HASH-TABLE*)
   (IL:FUNCTIONS ADD-ADVICE DELETE-ADVICE GET-ADVICE-MIDDLE-MAN SET-ADVICE-MIDDLE-MAN INSERT-ADVICE-FORM
    )
   (IL:SETFS GET-ADVICE-MIDDLE-MAN)
   ;;
   ;; Hacking the actual advice forms.
   (IL:FUNCTIONS CREATE-ADVISED-DEFINITION MAKE-AROUND-BODY)
   ;;
   ;; Dealing with the File Manager
   (IL:FILEPKGCOMS IL:ADVISE IL:ADVISE)
   (IL:FUNCTIONS XCL:REINSTALL-ADVICE)
   (IL:FUNCTIONS ADVICE-GETDEF ADVICE-PUTDEF ADVICE-DELDEF ADVICE-HASDEF ADVICE-NEWCOM
    ADVICE-FILE-DEFINITIONS ADVICE-CONTENTS ADVICE-ADDTOCOM)
   (IL:PROP IL:PROPTYPE IL:ADVISED)
   ;;
   ;; Dealing with old-style advice
   (IL:FUNCTIONS IL:READWISE1 ADD-OLD-STYLE-ADVICE CANONICALIZE-ADVICE-SYMBOL
    CANONICALIZE-ADVICE-WHEN-SPEC CANONICALIZE-ADVICE-WHERE-SPEC)
   (IL:DEFINE-TYPES XCL:ADVISED-FUNCTIONS)
   (IL:FUNCTIONS XCL:DEFADVICE)
   ;; Arrange for the proper package. Because of the DEFSTRUCT above, we must have the file dumped in the SYSTEM package.
   (IL:PROP (IL:MAKEFILE-ENVIRONMENT IL:FILETYPE)
    IL:ADVISE)
   (IL:DECLARE\ : IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY IL:COMPILELVARS (IL:ADDVARS (IL:NLAMA
    IL:READWISE
    IL:UNADVISE
    )
    (IL:NLAML)
    (IL:LAMA
    IL:ADVISE)
    )))

(DEFSTRUCT (ADVICE (:TYPE LIST))
  BEFORE
  AFTER
  AROUND)

(DEFVAR IL:ADVISED-FNS NIL)

(DEFVAR *UNADVISED-FNS* NIL)

;;
;; Interlisp entry points.
```

(IL:DEFINEQ

(IL:ADVISE

(IL:LAMBDA IL:ARGS

; Edited 6-Apr-87 18:00 by Pavel

::: ADVISE the FN given. ADVISE1 is for advice of the type (foo IN bar)

(LET (IL:FN IL:WHEN IL:WHERE IL:WHAT)

; First we straighten out the arguments given to us

(IL:SETQ IL:FN (IL:ARG IL:ARGS 1))

(CASE IL:ARGS

(2 (IL:SETQ IL:WHAT (IL:ARG IL:ARGS 2)))

(3 (IL:SETQ IL:WHEN (IL:ARG IL:ARGS 2))

(IL:SETQ IL:WHAT (IL:ARG IL:ARGS 3)))

(4 (IL:SETQ IL:WHEN (IL:ARG IL:ARGS 2))

(IL:SETQ IL:WHERE (IL:ARG IL:ARGS 3))

(IL:SETQ IL:WHAT (IL:ARG IL:ARGS 4)))

(T (IL:IF (< IL:ARGS 2)

IL:THEN (ERROR 'IL:TOO-FEW-ARGUMENTS :CALLEE 'IL:ADVISE :ACTUAL IL:ARGS :MINIMUM 2)

IL:ELSE (ERROR 'IL:TOO-MANY-ARGUMENTS :CALLEE 'IL:ADVISE :ACTUAL IL:ARGS :MAXIMUM 4))))

(IL:SETQ IL:WHEN (CANONICALIZE-ADVICE-WHEN-SPEC IL:WHEN))

(IL:SETQ IL:WHERE (CANONICALIZE-ADVICE-WHERE-SPEC IL:WHERE))

(IL:IF (IL:NLISTP IL:FN)

IL:THEN (XCL:ADVISE-FUNCTION IL:FN IL:WHAT :WHEN IL:WHEN :PRIORITY IL:WHERE)

IL:ELSEIF (IL:STRING-EQUAL (CADR IL:FN)

"IN")

IL:THEN (XCL:ADVISE-FUNCTION (FIRST IL:FN)

IL:WHAT :IN (THIRD IL:FN)

:WHEN IL:WHEN :PRIORITY IL:WHERE)

IL:ELSE (IL:FOR IL:X IL:IN IL:FN IL:JOIN (IL:IF (IL:NLISTP IL:X)

IL:THEN (XCL:ADVISE-FUNCTION IL:X IL:WHAT :WHEN IL:WHEN :PRIORITY IL:WHERE)

IL:ELSE (XCL:ADVISE-FUNCTION (FIRST IL:X)

IL:WHAT :IN (THIRD IL:X)

:WHEN IL:WHEN :PRIORITY IL:WHERE))))))

(IL:UNADVISE

(IL:NLAMBDA IL:FNS

; Edited 6-Apr-87 16:21 by Pavel

(IL:SETQ IL:FNS (IL:NLAMBDA.ARGs IL:FNS))

(FLET ((IL:UNADVISE-ENTRY (IL:ENTRY)

(IL:IF (IL:LISTP IL:ENTRY)

IL:THEN (XCL:UNADVISE-FUNCTION (FIRST IL:ENTRY)

:IN

(THIRD IL:ENTRY))

IL:ELSE (XCL:UNADVISE-FUNCTION IL:ENTRY))))

(COND

((NULL IL:FNS)

(IL:FOR IL:ENTRY IL:IN (IL:REVERSE IL:ADVISEDfNS) IL:JOIN (IL:UNADVISE-ENTRY IL:ENTRY)))

((IL:EQUAL IL:FNS '(T))

(AND (NOT (NULL IL:ADVISEDfNS))

(IL:UNADVISE-ENTRY (CAR IL:ADVISEDfNS))))

(T (IL:FOR IL:ENTRY IL:IN IL:FNS IL:JOIN (IL:UNADVISE-ENTRY IL:ENTRY))))))

(IL:READVISE

(IL:NLAMBDA IL:FNS

; Edited 6-Apr-87 16:52 by Pavel

(IL:SETQ IL:FNS (IL:NLAMBDA.ARGs IL:FNS))

(FLET ((IL:READVISE-ENTRY (IL:ENTRY)

(IL:IF (IL:LISTP IL:ENTRY)

IL:THEN (XCL:READVISE-FUNCTION (FIRST IL:ENTRY)

:IN

(THIRD IL:ENTRY))

IL:ELSE (XCL:READVISE-FUNCTION IL:ENTRY))))

(COND

((NULL IL:FNS)

(IL:FOR IL:ENTRY IL:IN (IL:REVERSE *UNADVISED-FNS*) IL:JOIN (IL:READVISE-ENTRY IL:ENTRY)))

; advise them all, in reverse order.

; simple case, advise just the last one that was unadvised.

((IL:EQUAL IL:FNS '(T))

(AND (NOT (NULL *UNADVISED-FNS*))

(IL:READVISE-ENTRY (CAR *UNADVISED-FNS))))

(T

; they gave us some functions, so advise THEM. We can't use

; READVISE-ENTRY here, because we may have to deal with

; old-style advice.

(IL:FOR IL:ENTRY IL:IN IL:FNS IL:JOIN (IL:READVISE1 IL:ENTRY))))))

)

(IL:PUTPROPS IL:ADVISE IL:ARGNAMES (IL:WHO IL:WHEN IL:WHERE IL:WHAT))

::

:: XCL entry points.

(DEFUN XCL:ADVISE-FUNCTION (XCL::FN-TO-ADVISE XCL::FORM &KEY (:IN XCL::IN-FN))

```

(WHEN :BEFORE)
(XCL::PRIORITY :LAST))
(COND
  ((CONSP XCL::FN-TO-ADVISE)
   (IL:FOR XCL::FN IL:IN XCL::FN-TO-ADVISE IL:JOIN (XCL:ADVISE-FUNCTION XCL::FN XCL::FORM :IN XCL::IN-FN :WHEN
   WHEN :PRIORITY XCL::PRIORITY)))
  ((CONSP XCL::IN-FN)
   (IL:FOR XCL::FN IL:IN XCL::IN-FN IL:JOIN (XCL:ADVISE-FUNCTION XCL::FN-TO-ADVISE XCL::FORM :IN XCL::FN :WHEN
   WHEN :PRIORITY XCL::PRIORITY)))
  ((NULL XCL::FORM)
   (FORMAT *ERROR-OUTPUT* "No advice given, so nothing done.")
   NIL)
  ((IL:UNSAFE.TO.MODIFY XCL::FN-TO-ADVISE "advise")
   (FORMAT *ERROR-OUTPUT* "~S not advised.~%" XCL::FN-TO-ADVISE)
   NIL)
  (T (COND
      (XCL::IN-FN (IF (NOT (HAS-CALLS XCL::IN-FN XCL::FN-TO-ADVISE))
                     (ERROR "~S is not called from ~S." XCL::FN-TO-ADVISE XCL::IN-FN)))
      (T (IF (NULL (IL:GETD XCL::FN-TO-ADVISE))
              (ERROR 'XCL:UNDEFINED-FUNCTION :NAME XCL::FN-TO-ADVISE))))
      (XCL:UNBREAK-FUNCTION XCL::FN-TO-ADVISE :IN XCL::IN-FN :NO-ERROR T)
      (COND
        ((NULL XCL::IN-FN)
         ;; Adjust the database of advice for this function.
         (WHEN (NOT (MEMBER XCL::FN-TO-ADVISE IL:ADVISEDFNS :TEST 'EQ))
                ; If FN-TO-ADVISE is not currently advised, the new advice
                ; replaces any that may have been given before.
              (DELETE-ADVICE XCL::FN-TO-ADVISE))
          (ADD-ADVICE XCL::FN-TO-ADVISE WHEN XCL::PRIORITY XCL::FORM)
          ;; Finish off the process. This part is shared with READWISE-FUNCTION.
          (FINISH-ADVISING XCL::FN-TO-ADVISE NIL))
        (T (LET* ((XCL::ADVICE-NAME `(,XCL::FN-TO-ADVISE :IN ,XCL::IN-FN))
                  (XCL::ALREADY-ADVISED? (MEMBER XCL::ADVICE-NAME IL:ADVISEDFNS :TEST 'EQUAL)))
              ;; Adjust the database of advice for this request.
              (WHEN (NOT XCL::ALREADY-ADVISED?)
                     ; If not currently advised, the new advice replaces any that may
                     ; have been given before.
                   (DELETE-ADVICE XCL::ADVICE-NAME))
                (ADD-ADVICE XCL::ADVICE-NAME WHEN XCL::PRIORITY XCL::FORM)
                ;; Finish off the process. This part is shared with READWISE-FUNCTION.
                (FINISH-ADVISING XCL::FN-TO-ADVISE XCL::IN-FN))))))
      (DEFUN XCL:UNADVISE-FUNCTION (XCL::FN-TO-UNADVISE &KEY (:IN XCL::IN-FN)
                                   XCL::NO-ERROR)
      (COND
        ((CONSP XCL::FN-TO-UNADVISE)
         (IL:FOR XCL::FN IL:IN XCL::FN-TO-UNADVISE IL:JOIN (XCL:UNADVISE-FUNCTION XCL::FN :IN XCL::IN-FN)))
        ((CONSP XCL::IN-FN)
         (IL:FOR XCL::FN IL:IN XCL::IN-FN IL:JOIN (XCL:UNADVISE-FUNCTION XCL::FN-TO-UNADVISE :IN XCL::FN)))
        (T (XCL:UNBREAK-FUNCTION XCL::FN-TO-UNADVISE :IN XCL::IN-FN :NO-ERROR T)
           (IF (NULL XCL::IN-FN)
               (LET ((XCL::ORIGINAL (GET XCL::FN-TO-UNADVISE 'IL:ADVISED)))
                   (COND
                     ((NULL XCL::ORIGINAL)
                      (UNLESS XCL::NO-ERROR (FORMAT *ERROR-OUTPUT* "~S is not advised.~%" XCL::FN-TO-UNADVISE))
                      NIL)
                     (T (IL:PUTD XCL::FN-TO-UNADVISE (IL:GETD XCL::ORIGINAL)
                               T)
                        (REMPROP XCL::FN-TO-UNADVISE 'IL:ADVISED)
                        (PUSH XCL::FN-TO-UNADVISE *UNADVISED-FNS*)
                        (SETQ IL:ADVISEDFNS (DELETE XCL::FN-TO-UNADVISE IL:ADVISEDFNS))
                        (LIST XCL::FN-TO-UNADVISE))))
               (LET* ((XCL::ADVICE-NAME `(,XCL::FN-TO-UNADVISE :IN ,XCL::IN-FN))
                      (XCL::MIDDLE-MAN (GET-ADVICE-MIDDLE-MAN XCL::ADVICE-NAME)))
                   (COND
                     ((NULL XCL::MIDDLE-MAN)
                      (UNLESS XCL::NO-ERROR (FORMAT *ERROR-OUTPUT* "~S is not advised.~%" XCL::ADVICE-NAME))
                      NIL)
                     (T (CHANGE-CALLS XCL::MIDDLE-MAN XCL::FN-TO-UNADVISE XCL::IN-FN)
                        (FINISH-UNADVISING XCL::ADVICE-NAME XCL::MIDDLE-MAN)
                        (LIST XCL::ADVICE-NAME))))))
               (DEFUN XCL:READWISE-FUNCTION (XCL::FN-TO-READWISE &KEY (:IN XCL::IN-FN))
               (COND
                 ((CONSP XCL::FN-TO-READWISE)
                  (IL:FOR XCL::FN IL:IN XCL::FN-TO-READWISE IL:JOIN (XCL:READWISE-FUNCTION XCL::FN :IN XCL::IN-FN)))
                 ((CONSP XCL::IN-FN)
                  (IL:FOR XCL::FN IL:IN XCL::IN-FN IL:JOIN (XCL:READWISE-FUNCTION XCL::FN-TO-READWISE :IN XCL::FN)))
                 (T (XCL:UNBREAK-FUNCTION XCL::FN-TO-READWISE :IN XCL::IN-FN :NO-ERROR T)
                    (FINISH-ADVISING XCL::FN-TO-READWISE XCL::IN-FN))))

```

```

(DEFUN UNADVISE-FROM-RESTORE-CALLS (FROM TO FN)
  (LET ((ENTRY (FIND-IF #'(LAMBDA (ENTRY)
    (AND (CONSP ENTRY)
          (EQ (FIRST ENTRY)
              FROM)
          (EQ (THIRD ENTRY)
              FN))))
        IL:ADVISEDFNS)))
  (ASSERT (NOT (NULL ENTRY))
    NIL "BUG: Inconsistency in SI::UNADVISE-FROM-RESTORE-CALLS")
  (FINISH-UNADVISING ENTRY TO)
  (FORMAT *TERMINAL-IO* "~S unadvised.~%" ENTRY)))

(DEFUN FINISH-ADVISING (FN-TO-ADVISE IN-FN)
  (COND
    ((NULL IN-FN)
     (LET* ((ALREADY-ADVISED? (MEMBER FN-TO-ADVISE IL:ADVISEDFNS :TEST 'EQ))
            (ORIGINAL (IF ALREADY-ADVISED?
                          (GET FN-TO-ADVISE 'IL:ADVISED)
                          (LET ((*PRINT-CASE* :UPCASE))
                            (MAKE-SYMBOL (FORMAT NIL "Original ~A" FN-TO-ADVISE))))))
      ;; Adjust the database of advice for this function.
      (WHEN (NOT ALREADY-ADVISED?)
        (IL:PUTD ORIGINAL (IL:GETD FN-TO-ADVISE)
          T))
        (IL:PUTD FN-TO-ADVISE (COMPILE NIL (CREATE-ADVISED-DEFINITION FN-TO-ADVISE ORIGINAL FN-TO-ADVISE)))
        (WHEN (NOT ALREADY-ADVISED?)
          (SETF (GET FN-TO-ADVISE 'IL:ADVISED)
            ORIGINAL)))
      ;; These are outside the WHEN because COMPILE calls VIRGINFN, which may unadvise the function.
      (SETQ *UNADVISED-FNS* (DELETE FN-TO-ADVISE *UNADVISED-FNS* :TEST 'EQ))
      (SETQ IL:ADVISEDFNS
        (CONS FN-TO-ADVISE (DELETE FN-TO-ADVISE IL:ADVISEDFNS :TEST 'EQ)))
      (IL:MARKASCHANGED FN-TO-ADVISE 'IL:ADVICE)
      (LIST FN-TO-ADVISE)))
    (T (LET* ((ADVICE-NAME `(,FN-TO-ADVISE :IN ,IN-FN))
              (ALREADY-ADVISED? (MEMBER ADVICE-NAME IL:ADVISEDFNS :TEST 'EQUAL))
              MIDDLE-MAN)
      ;; Create a middle-man for this request. If one has already been created, use it.
      (SETQ MIDDLE-MAN (OR (GET-ADVICE-MIDDLE-MAN ADVICE-NAME)
        (SETF (GET-ADVICE-MIDDLE-MAN ADVICE-NAME)
          (CONSTRUCT-MIDDLE-MAN FN-TO-ADVISE IN-FN))))
      ;; Give the middle-man the new advised definition.
      (IL:PUTD MIDDLE-MAN (COMPILE NIL (CREATE-ADVISED-DEFINITION FN-TO-ADVISE FN-TO-ADVISE ADVICE-NAME
        )))
      (WHEN (NOT ALREADY-ADVISED?)
        ;; Redirect any calls to FN-TO-ADVISE in IN-FN to call the middle-man.
        (CHANGE-CALLS FN-TO-ADVISE MIDDLE-MAN IN-FN 'UNADVISE-FROM-RESTORE-CALLS))
      ;; Save a trail of information. These are outside the WHEN because COMPILE calls VIRGINFN, which may unadvise the function.
      (SETQ *UNADVISED-FNS* (DELETE ADVICE-NAME *UNADVISED-FNS* :TEST 'EQUAL))
      (SETQ IL:ADVISEDFNS
        (CONS ADVICE-NAME (DELETE ADVICE-NAME IL:ADVISEDFNS :TEST 'EQUAL)))
      (IL:MARKASCHANGED ADVICE-NAME 'IL:ADVICE)
      (LIST ADVICE-NAME))))))

(DEFUN FINISH-UNADVISING (ADVICE-NAME MIDDLE-MAN)
  (SETQ IL:ADVISEDFNS (DELETE ADVICE-NAME IL:ADVISEDFNS :TEST 'EQUAL))
  (PUSH ADVICE-NAME *UNADVISED-FNS*))

;;
;; The advice database.

(DEFVAR *ADVICE-HASH-TABLE* (MAKE-HASH-TABLE :TEST 'EQUAL))
;;; Hash-table mapping either a function name or a list in the form (FOO :IN BAR) to a pair (advice . middle-man).

)

(DEFUN ADD-ADVICE (NAME WHEN PRIORITY FORM)
  ;; Advice is stored on the hash table SI::*ADVICE-HASH-TABLE*. It is actually stored as a cons whose CAR is the advice and CDR is the middle-man
  ;; name (for advice of the type (FOO :IN BAR)).
  (LET* ((OLD-ADVICE (GETHASH NAME *ADVICE-HASH-TABLE*))

```

```

(ADVICE (IF (NULL OLD-ADVICE)
            (MAKE-ADVICE)
            (CAR OLD-ADVICE))))
(ECASE WHEN
  (:BEFORE (SETF (ADVICE-BEFORE ADVICE)
                 (INSERT-ADVICE-FORM FORM PRIORITY (ADVICE-BEFORE ADVICE))))
  (:AFTER (SETF (ADVICE-AFTER ADVICE)
                (INSERT-ADVICE-FORM FORM PRIORITY (ADVICE-AFTER ADVICE))))
  (:AROUND (SETF (ADVICE-AROUND ADVICE)
                 (INSERT-ADVICE-FORM FORM PRIORITY (ADVICE-AROUND ADVICE))))))
(WHEN (NULL OLD-ADVICE)
  (SETF (GETHASH NAME *ADVICE-HASH-TABLE*)
        (CONS ADVICE NIL))))

```

```

(DEFUN DELETE-ADVICE (NAME)
  (REMHASH NAME *ADVICE-HASH-TABLE*))

```

```

(DEFUN GET-ADVICE-MIDDLE-MAN (NAME)
  (CDR (GETHASH NAME *ADVICE-HASH-TABLE*)))

```

```

(DEFUN SET-ADVICE-MIDDLE-MAN (NAME MIDDLE-MAN)
  (SETF (CDR (GETHASH NAME *ADVICE-HASH-TABLE*))
        MIDDLE-MAN))

```

```

(DEFUN INSERT-ADVICE-FORM (FORM PRIORITY ENTRY-LIST)

```

;;; Insert the new advice FORM into ENTRY-LIST using PRIORITY as a specification of where in that list to put it. If an equalish piece of advice already exists, remove it first.

```

(LET ((ENTRY (LIST PRIORITY FORM)))
  (SETF ENTRY-LIST
        (LABELS ((EQUALISH
                  (X Y)
                  ;; EQUALP, but don't ignore case in strings.
                  (TYPECASE X
                    (SYMBOL (EQ X Y))
                    (CONS (AND (CONSP Y)
                               (EQUALISH (CAR X)
                                           (CAR Y))
                               (EQUALISH (CDR X)
                                           (CDR Y))))))
                    (NUMBER (AND (NUMBERP Y)
                                   (= X Y)))
                    (CHARACTER (AND (CHARACTERP Y)
                                      (CHAR= X Y)))
                    (STRING (AND (STRINGP Y)
                                   (STRING= X Y)))
                    (PATHNAME (AND (PATHNAMEP Y)
                                    (IL:%PATHNAME-EQUAL X Y)))
                    (VECTOR (AND (VECTORP Y)
                                   (LET ((SX (LENGTH X))
                                         (AND (EQL SX (LENGTH Y))
                                               (DOTIMES (I SX T)
                                                         (IF (NOT (EQUALISH (AREF X I)
                                                                           (AREF Y I)))
                                                             (RETURN NIL)))))))
                    (ARRAY (AND (ARRAYP Y)
                                   (EQUAL (ARRAY-DIMENSIONS X)
                                           (ARRAY-DIMENSIONS Y))
                                   (LET ((FX (IL:%FLATTEN-ARRAY X))
                                         (FY (IL:%FLATTEN-ARRAY Y)))
                                     (DOTIMES (I (ARRAY-TOTAL-SIZE X)
                                               T)
                                       (IF (NOT (EQUALISH (AREF FX I)
                                                             (AREF FY I)))
                                           (RETURN NIL)))))))
                    (T ;; so that datatypes will be properly compared
                     (OR (EQ X Y)
                         (LET ((TYPENAME (IL:TYPENAME X))
                               (AND (EQ TYPENAME (IL:TYPENAME Y))
                                   (LET ((DESCRIPTORS (IL:GETDESCRIPTORS TYPENAME)))
                                       (IF DESCRIPTORS
                                           (IL:FOR FIELD IL:IN DESCRIPTORS
                                             IL:ALWAYS (EQUALISH (IL:FFETCHFIELD FIELD X)
                                                                (IL:FFETCHFIELD FIELD Y))))))))))
                    (DELETE-IF #'(LAMBDA (OLD-ENTRY)
                                (XCL:DESTRUCTURING-BIND (OLD-PRIORITY OLD-FORM)
                                                         OLD-ENTRY
                                                         (AND (EQUAL PRIORITY OLD-PRIORITY)
                                                             (EQUALISH FORM OLD-FORM))))
                              ENTRY-LIST)))

```

```
(COND
  ((NULL ENTRY-LIST)
   (LIST ENTRY))
  ((EQ PRIORITY :FIRST)
   (CONS ENTRY ENTRY-LIST))
  ((EQ PRIORITY :LAST)
   (NCONC ENTRY-LIST (LIST ENTRY)))
  (T
   (UNLESS (AND (CONSP PRIORITY)
                (MEMBER (CAR PRIORITY)
                        ' (IL:BEFORE IL:AFTER)))
            (ERROR "Malformed priority argument to ADVISE: ~S" PRIORITY))
    (XCL:CONDITION-CASE (IL:EDITE ENTRY-LIST `((IL:LC ,@(CDR PRIORITY))
                                                (IL:BELOW IL:^)
                                                (, (CAR PRIORITY)
                                                  ,ENTRY)))
                       (ERROR (C)
                              (ERROR "Error from EDITE during insertion of new advice:~% ~A~%" C)))
    ENTRY-LIST))))
```

; PRIORITY is a command to the old TTY Editor.

```
(DEFSETF GET-ADVICE-MIDDLE-MAN SET-ADVICE-MIDDLE-MAN)
```

```
::
;; Hacking the actual advice forms.
```

```
(DEFUN CREATE-ADVISED-DEFINITION (ADVISED-FN FN-TO-CALL ADVICE-NAME)
  (MULTIPLE-VALUE-BIND (LAMBDA-CAR ARG-LIST CALLING-FORM)
    (FUNCTION-WRAPPER-INFO ADVISED-FN FN-TO-CALL)
    (LET* ((ADVISE (CAR (GETHASH ADVICE-NAME *ADVICE-HASH-TABLE*)))
           (BEFORE-FORMS (MAPCAR 'SECOND (ADVISE-BEFORE ADVISE)))
           (AFTER-FORMS (MAPCAR 'SECOND (ADVISE-AFTER ADVISE)))
           (AROUND-FORMS (MAPCAR 'SECOND (ADVISE-AROUND ADVISE)))
           (BODY-FORM (MAKE-AROUND-BODY CALLING-FORM AROUND-FORMS)))
          `(,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 ' (:ADVISED ,ADVICE-NAME))
          (BLOCK NIL
            (XCL:DESTRUCTURING-BIND (IL:!VALUE . IL:!OTHER-VALUES)
              (MULTIPLE-VALUE-LIST (PROGN ,@BEFORE-FORMS ,BODY-FORM)
                ,@AFTER-FORMS
                (APPLY 'VALUES IL:!VALUE IL:!OTHER-VALUES))))))))
```

```
(DEFUN MAKE-AROUND-BODY (CALLING-FORM AROUND-FORMS)
  (REDUCE #'(LAMBDA (CURRENT-BODY NEXT-AROUND-FORM)
            (LET ((CANONICALIZED-AROUND-FORM (SUBST ' (XCL:INNER)
                                                    ' IL:* NEXT-AROUND-FORM)))
              `(MACROLET ((XCL:INNER NIL ' ,CURRENT-BODY)
                          ,CANONICALIZED-AROUND-FORM))
                AROUND-FORMS :INITIAL-VALUE CALLING-FORM)))
  AROUND-FORMS :INITIAL-VALUE CALLING-FORM))
```

```
::
;; Dealing with the File Manager
```

```
(IL:PUTDEF 'IL:ADVICE 'IL:FILEPKGCOMS
  '((IL:COM IL:MACRO (IL:X (IL:P IL:* (ADVISE-FILE-DEFINITIONS 'IL:X NIL)))
    IL:CONTENTS IL:NILL IL:ADD ADVICE-ADDTOCOM)
  (TYPE IL:DESCRIPTION "advice" IL:NEWCOM ADVISE-NEWCOM IL:GETDEF ADVISE-GETDEF IL:DELDEF ADVISE-DELDEF
  IL:PUTDEF ADVISE-PUTDEF IL:HASDEF ADVISE-HASDEF)))

(IL:PUTDEF 'IL:ADVICE 'IL:FILEPKGCOMS '((IL:COM IL:MACRO (IL:X (IL:P IL:* (ADVISE-FILE-DEFINITIONS 'IL:X T)))
  IL:CONTENTS ADVISE-CONTENTS IL:ADD ADVICE-ADDTOCOM)))
```

```
(DEFUN XCL:REINSTALL-ADVICE (XCL::NAME &KEY XCL::BEFORE XCL::AFTER XCL::AROUND)
  (IL:FOR XCL::ADVICE IL:IN XCL::BEFORE IL:DO (XCL:DESTRUCTURING-BIND (XCL::PRIORITY XCL::FORM)
    XCL::ADVICE
    (ADD-ADVICE XCL::NAME :BEFORE XCL::PRIORITY XCL::FORM)))
  (IL:FOR XCL::ADVICE IL:IN XCL::AFTER IL:DO (XCL:DESTRUCTURING-BIND (XCL::PRIORITY XCL::FORM)
    XCL::ADVICE
    (ADD-ADVICE XCL::NAME :AFTER XCL::PRIORITY XCL::FORM)))
  (IL:FOR XCL::ADVICE IL:IN XCL::AROUND IL:DO (XCL:DESTRUCTURING-BIND (XCL::PRIORITY XCL::FORM)
    XCL::ADVICE
    (ADD-ADVICE XCL::NAME :AROUND XCL::PRIORITY XCL::FORM))))
```

```
(DEFUN ADVISE-GETDEF (NAME TYPE OPTIONS)
  (LET ((ADVISE (CAR (GETHASH NAME *ADVICE-HASH-TABLE*))))
```

```
(AND ADVICE (APPEND (IL:FOR ENTRY IL:IN (ADVICE-BEFORE ADVICE) IL:COLLECT (CONS ' :BEFORE (COPY-TREE ENTRY)
))
(IL:FOR ENTRY IL:IN (ADVICE-AFTER ADVICE) IL:COLLECT (CONS ' :AFTER (COPY-TREE ENTRY)))
(IL:FOR ENTRY IL:IN (ADVICE-AROUND ADVICE) IL:COLLECT (CONS ' :AROUND (COPY-TREE ENTRY)
))))))
```

```
(DEFUN ADVICE-PUTDEF (NAME TYPE DEFINITION)
  (LET ((CANONICAL-DEFN (IL:FOR ENTRY IL:IN DEFINITION IL:COLLECT (LIST (CANONICALIZE-ADVICE-WHEN-SPEC
(CAR ENTRY))
(CANONICALIZE-ADVICE-WHERE-SPEC
(COPY-TREE (CADR ENTRY)))
(COPY-TREE (CADDR ENTRY))))))
    (CURRENT-ADVICE (OR (CAR (GETHASH NAME *ADVICE-HASH-TABLE*))
(CAR (SETF (GETHASH NAME *ADVICE-HASH-TABLE*)
(CONS (MAKE-ADVICE)
NIL))))))
    (SETF (ADVICE-BEFORE CURRENT-ADVICE)
(MAPCAR #'REST (IL:FOR ENTRY IL:IN CANONICAL-DEFN IL:WHEN (EQ (CAR ENTRY)
:BEFORE)
IL:COLLECT ENTRY)))
    (SETF (ADVICE-AFTER CURRENT-ADVICE)
(MAPCAR #'REST (IL:FOR ENTRY IL:IN CANONICAL-DEFN IL:WHEN (EQ (CAR ENTRY)
:AFTER)
IL:COLLECT ENTRY)))
    (SETF (ADVICE-AROUND CURRENT-ADVICE)
(MAPCAR #'REST (IL:FOR ENTRY IL:IN CANONICAL-DEFN IL:WHEN (EQ (CAR ENTRY)
:AROUND)
IL:COLLECT ENTRY)))
    (IF (CONSP NAME)
(XCL:READWISE-FUNCTION (FIRST NAME)
:IN
(THIRD NAME))
(XCL:READWISE-FUNCTION NAME))))))
```

```
(DEFUN ADVICE-DELDEF (NAME TYPE)
  (DECLARE (IGNORE TYPE))
  (WHEN (MEMBER NAME IL:ADVISED FNS :TEST 'EQUAL)
    (IF (CONSP NAME)
      (XCL:UNADVISE-FUNCTION (FIRST NAME)
        :IN
        (THIRD NAME))
      (XCL:UNADVISE-FUNCTION NAME))
    (FORMAT *TERMINAL-IO* "~S unadvised." NAME)
    (REMHASH NAME *ADVICE-HASH-TABLE*)))
```

```
(DEFUN ADVICE-HASDEF (NAME TYPE SOURCE)
  (AND (GETHASH NAME *ADVICE-HASH-TABLE*)
    (OR NAME T)))
```

```
(DEFUN ADVICE-NEWCOM (NAME TYPE LISTNAME FILE)
```

;; If you make a new com for ADVICE, you should make an ADVISE command.

```
(IL:DEFAULTMAKENEWCOM NAME 'IL:ADVISE LISTNAME FILE))
```

```
(DEFUN ADVICE-FILE-DEFINITIONS (NAMES READWISE?)
```

;; READWISE? is true for the File Manager command ADVISE and false for the command ADVICE. For ADVICE, we want to emit a form to readwise the
;; named functions after reinstalling the advice.

```
(LET ((REAL-NAMES NIL)
      `(@ (IL:FOR FN IL:IN NAMES
          IL:COLLECT (LET* ((NAME (IL:IF (CONSP FN)
                                        IL:THEN (ASSERT (AND (EQ (SECOND FN)
                                                                :IN)
                                                                (= 3 (LENGTH FN)))
                                                                NIL "~S should be of the form (FOO :IN BAR)" FN)
                                        FN
                                        IL:ELSE (LET ((NAME (CANONICALIZE-ADVICE-SYMBOL FN))
                                                       (OLD-ADVICE (GET FN 'IL:READWISE)))
                                                       (WHEN OLD-ADVICE
                                                         (ADD-OLD-STYLE-ADVICE NAME OLD-ADVICE)
                                                         (REMPROP FN 'IL:READWISE))
                                                       NAME))))
          (ADVICE (CAR (GETHASH NAME *ADVICE-HASH-TABLE*))))
      (ASSERT (NOT (NULL ADVICE))
        NIL "Can't find advice for ~S" NAME)
      (PUSH NAME REAL-NAMES)
      ` (XCL:REINSTALL-ADVICE ' ,NAME
        ,@ (AND (ADVICE-BEFORE ADVICE)
                ` (:BEFORE ' , (ADVICE-BEFORE ADVICE)))
        ,@ (AND (ADVICE-AFTER ADVICE)
```

```

\ (:AFTER ' , (ADVICE-AFTER ADVICE)))
, @ (AND (ADVICE-AROUND ADVICE)
\ (:AROUND ' , (ADVICE-AROUND ADVICE))))))
, @ (AND READWISE? \ (:IL:READWISE , @ (REVERSE REAL-NAMES))))))

```

```

(DEFUN ADVICE-CONTENTS (COM NAME TYPE)
  (AND (EQ TYPE 'IL:ADVICE)
    (COND
      ((NULL NAME) ; Return a list of the ADVICE's in the given COM.
       (CDR COM))
      ((EQ NAME 'T) ; Return T if there are ANY ADVICE's in the given COM.
       (NOT (NULL (CDR COM))))
      ((OR (SYMBOLP NAME)
           (= (LENGTH NAME)
              3)
           (EQ (SECOND NAME)
                :IN)) ; Return T iff an ADVICE named NAME in the given COM.
       (AND (MEMBER NAME (CDR COM)
                     :TEST
                     'EQUAL)
            T))
      (T) ; NAME is a list of names. Return the intersection of that list with
           ; the ADVICE's in the given COM.
      (INTERSECTION NAME (CDR COM)
                     :TEST
                     'EQUAL))))))

```

```

(DEFUN ADVICE-ADDTOCOM (COM NAME TYPE NEAR)
  ;; This is the ADD method for both of the ADVICE and ADVISE commands.

```

```

  ;; Add the given name only if the type is ADVICE. Also, add it to ADVICE commands only if a NEAR was specified. We want to normally create only
  ;; ADVISE commands. If the user really wants an ADVICE command, they'll have to create it themselves.

```

```

  (AND (EQ TYPE 'IL:ADVICE)
        (OR (EQ (CAR COM)
                 'IL:ADVISE)
            (NOT (NULL NEAR))))
    (IL:ADDTOCOM1 COM NAME NEAR NIL))

(IL:PUTPROPS IL:ADVISED IL:PROPTYPE IGNORE)

```

```

;;
;; Dealing with old-style advice

```

```

(DEFUN IL:READWISE1 (IL:FN)
  (FLET ((IL:READWISE-ENTRY (IL:ENTRY)
        (IL:IF (IL:LISTP IL:ENTRY)
                IL:THEN (XCL:READWISE-FUNCTION (FIRST IL:ENTRY)
                                                :IN
                                                (THIRD IL:ENTRY))
                IL:ELSE (XCL:READWISE-FUNCTION IL:ENTRY))))
    (IL:IF (IL:LISTP IL:FN)
            IL:THEN (ASSERT (IL:STRING-EQUAL (SECOND IL:FN)
                                             "IN")
                           NIL "~S should be in the form (FOO IN BAR).~%" IL:FN)
              (IL:READWISE-ENTRY IL:FN))
            IL:ELSE (LET ((IL:NAME (CANONICALIZE-ADVICE-SYMBOL IL:FN))
                          (IL:OLD-ADVICE (GET IL:FN 'IL:READWISE)))
                      (IL:IF IL:OLD-ADVICE
                              IL:THEN (ADD-OLD-STYLE-ADVICE IL:NAME IL:OLD-ADVICE)
                              (REMPROP IL:FN 'IL:READWISE))
                      (IL:READWISE-ENTRY IL:NAME))))))

```

```

(DEFUN ADD-OLD-STYLE-ADVICE (NAME OLD-ADVICE)

```

```

  ;; OLD-ADVICE should the value of the READWISE property of some symbol. Note that the CAR of that value is the old middle-man used for -IN-
  ;; advice. Thus, we take the CDR below.

```

```

  (WHEN (NOT (MEMBER NAME IL:ADVISEDFN :TEST 'EQUAL))
    (DELETE-ADVICE NAME))
  (IL:FOR ADVICE IL:IN (CDR OLD-ADVICE) IL:DO (XCL:DESTRUCTURING-BIND (WHEN WHERE WHAT)
                                                                    ADVICE
                                                                    ;; Translate Interlisp names to the new standard.
                                                                    (ADD-ADVICE NAME (CANONICALIZE-ADVICE-WHEN-SPEC WHEN)
                                                                    (CANONICALIZE-ADVICE-WHERE-SPEC WHERE)
                                                                    WHAT))))

```

```

(DEFUN CANONICALIZE-ADVICE-SYMBOL (SYMBOL)
  (LET ((IN-POS (IL:STRPOS "-IN-" SYMBOL)))

```



```
(IF (NULL IN-POS)
  SYMBOL
  (LIST (IL:SUBATOM SYMBOL 1 (1- IN-POS))
        :IN
        (IL:SUBATOM SYMBOL (+ IN-POS 4)
                            NIL))))))
```

```
(DEFUN CANONICALIZE-ADVICE-WHEN-SPEC (SPEC)
  (IF (NULL SPEC)
    ' :BEFORE
    (INTERN (STRING SPEC)
            "KEYWORD"))))
```

```
(DEFUN CANONICALIZE-ADVICE-WHERE-SPEC (SPEC)
  (CASE SPEC
    ((NIL LAST IL:BOTTOM IL:END :LAST) ' :LAST)
    ((IL:TOP IL:FIRST :FIRST) ' :FIRST)
    (T (IF (CONSP SPEC)
           SPEC
           (ERROR "Illegal WHERE specification to ADVISE: ~S" SPEC))))))
```

```
(XCL:DEF-DEFINE-TYPE XCL:ADVISED-FUNCTIONS "Advised function definitions")
```

```
(XCL:DEFDEFINER (XCL:DEFADVICE (:PROTOTYPE (LAMBDA (XCL::NAME)
                                             \ (XCL:DEFADVICE ,XCL::NAME
                                                "advice"))))
  XCL:ADVISED-FUNCTIONS (XCL::NAME &BODY XCL::ADVICE-FORMS)
  \ (PROGN ,. (XCL:WITH-COLLECTION
              (DOLIST (XCL::ADVICE XCL::ADVICE-FORMS)
                (XCL:COLLECT (XCL:DESTRUCTURING-BIND
                              (XCL::FN-TO-ADVISE XCL::FORM &KEY XCL::IN WHEN XCL::PRIORITY)
                              XCL::ADVICE
                              \ (XCL:ADVISE-FUNCTION ' ,XCL::FN-TO-ADVISE ' ,XCL::FORM
                                  ,@ (AND XCL::IN `(:IN ' ,XCL::IN))
                                  ,@ (AND WHEN `(:WHEN ,WHEN))
                                  ,@ (AND XCL::PRIORITY `(:PRIORITY ,XCL::PRIORITY))))))))))
```

:: Arrange for the proper package. Because of the DEFSTRUCT above, we must have the file dumped in the SYSTEM package.

```
(IL:PUTPROPS IL:ADVISE IL:MAKEFILE-ENVIRONMENT (:READTABLE "XCL" :PACKAGE "SYSTEM"))
(IL:PUTPROPS IL:ADVISE IL:FILETYPE :COMPILE-FILE)
(IL:DECLARE\ : IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY IL:COMPILERVERS
(IL:ADDTOVAR IL:NLAMA IL:READWISE IL:UNADVISE)
(IL:ADDTOVAR IL:NLAML )
(IL:ADDTOVAR IL:LAMA IL:ADVISE)
)
(IL:PUTPROPS IL:ADVISE IL:COPYRIGHT ("Venue & Xerox Corporation" T 1978 1984 1986 1987 1988 1990))
```

FUNCTION INDEX

| | | | | | |
|-------------------------------|---|--------------------------------------|---|-----------------------------------|---|
| ADD-ADVICE | 4 | ADVISE-CONTENTS | 8 | INSERT-ADVICE-FORM | 5 |
| ADD-OLD-STYLE-ADVICE | 8 | XCL:ADVISE-FUNCTION | 2 | MAKE-AROUND-BODY | 6 |
| ADVICE-ADDTOCOM | 8 | CANONICALIZE-ADVICE-SYMBOL | 8 | IL:READVICE | 2 |
| ADVICE-DELDEF | 7 | CANONICALIZE-ADVICE-WHEN-SPEC | 9 | XCL:READVICE-FUNCTION | 3 |
| ADVICE-FILE-DEFINITIONS | 7 | CANONICALIZE-ADVICE-WHERE-SPEC | 9 | IL:READVICE1 | 8 |
| ADVICE-GETDEF | 6 | CREATE-ADVISED-DEFINITION | 6 | XCL:REINSTALL-ADVICE | 6 |
| ADVICE-HASDEF | 7 | DELETE-ADVICE | 5 | SET-ADVICE-MIDDLE-MAN | 5 |
| ADVICE-NEWCOM | 7 | FINISH-ADVISING | 4 | IL:UNADVISE | 2 |
| ADVICE-PUTDEF | 7 | FINISH-UNADVISING | 4 | UNADVISE-FROM-RESTORE-CALLS | 4 |
| IL:ADVISE | 2 | GET-ADVICE-MIDDLE-MAN | 5 | XCL:UNADVISE-FUNCTION | 3 |

VARIABLE INDEX

| | | | | | |
|---------------------------|---|-----------------------|---|----------------------|---|
| *ADVISE-HASH-TABLE* | 4 | *UNADVISED-FNS* | 1 | IL:ADVISED-FNS | 1 |
|---------------------------|---|-----------------------|---|----------------------|---|

PROPERTY INDEX

| | | | |
|-----------------|-----|------------------|---|
| IL:ADVISE | 2,9 | IL:ADVISED | 8 |
|-----------------|-----|------------------|---|

DEFINER INDEX

| | |
|---------------------|---|
| XCL:DEFADVISE | 9 |
|---------------------|---|

DEFINE-TYPE INDEX

| | |
|-----------------------------|---|
| XCL:ADVISED-FUNCTIONS | 9 |
|-----------------------------|---|

SETF INDEX

| | |
|-----------------------------|---|
| GET-ADVICE-MIDDLE-MAN | 6 |
|-----------------------------|---|

STRUCTURE INDEX

| | |
|--------------|---|
| ADVISE | 1 |
|--------------|---|
