File created: 30-Apr-2023 13:54:00 **{DSK}<home>larry>il>medley>library>WHERE-IS.;2**

edit by: lmm

changes to: (IL:FUNCTIONS ADD-WHERE-IS-DATABASE)

previous date: 11-Mar-2022 22:40:32 {DSK}<home>larry>il>medley>library>WHERE-IS.;1

Read Table: XCL

Package: XEROX-COMMON-LISP

Format: XCCS

```
(IL:RPAQQ IL:WHERE-ISCOMS
          ((IL:P (PROVIDE "WHERE-IS")
                 (REQUIRE "HASH-FILE" "HASH-FILE.DFASL")
                 (REQUIRE "CASH-FILE" "CASH-FILE.DFASL"))
           (IL:COMS
                  ;; run time code
                  (IL:FUNCTIONS HASH-FILE-WHERE-IS HASH-FILE-TYPES-OF GET-WHERE-IS-ENTRIES WHERE-IS-READ-FN
                         ADD-WHERE-IS-DATABASES ADD-WHERE-IS-DATABASE DEL-WHERE-IS-DATABASE
                         SAME-WHERE-IS-DATABASE CLOSE-WHERE-IS-FILES)
                  (IL:ADDVARS (IL:AROUNDEXITFNS CLOSE-WHERE-IS-FILES))
                  (IL:VARIABLES *WHERE-IS-CASH-FILES* *WHERE-IS-CASH-SIZE*))
           (IL:COMS
                  ;; notice time code
                  (IL:FUNCTIONS WHERE-IS-NOTICE WHERE-IS-NOTICE-INTERNAL WHERE-IS-FILES
                         WHERE-IS-DEFAULT-DEFINE-TYPES WHERE-IS-NAMESTRING WHERE-IS-READ-COMS
                         WHERE-IS-SET-WRITE-DATE WHERE-IS-GET-WRITE-DATE)
                  (IL:VARIABLES *WHERE-IS-HASH-FILE-SIZE* *WHERE-IS-IGNORE-DEFINE-TYPES*))
           (IL:PROP (IL:MAKEFILE-ENVIRONMENT IL:FILETYPE)
                  IL:WHERE-IS)))

(PROVIDE "WHERE-IS")

(REQUIRE "HASH-FILE" "HASH-FILE.DFASL")

(REQUIRE "CASH-FILE" "CASH-FILE.DFASL")

;; run time code

(DEFUN HASH-FILE-WHERE-IS (NAME TYPE)
   ;; return a list of file names containing NAME of TYPE
   (REMOVE-DUPLICATES (MAPCAN #'(LAMBDA (ENTRY)
                                  (CDR (ASSOC TYPE ENTRY)))
                              (GET-WHERE-IS-ENTRIES NAME))
          :TEST
          'STRING=))

(DEFUN HASH-FILE-TYPES-OF (NAME &OPTIONAL (POSSIBLE-TYPES (WHERE-IS-DEFAULT-DEFINE-TYPES)))
   (LET ((ENTRIES (GET-WHERE-IS-ENTRIES NAME))
         (TYPES NIL))
        (DOLIST (TYPE POSSIBLE-TYPES)
            (DOLIST (ENTRY ENTRIES)
                (WHEN (ASSOC TYPE ENTRY)
                    (PUSH TYPE TYPES)
                    (RETURN))))
        (REMOVE-DUPLICATES TYPES)))

(DEFUN GET-WHERE-IS-ENTRIES (NAME)
   ;; return a list of all entries for name in *WHERE-IS-CASH-FILES*
   (MAPLIST #'(LAMBDA (TAIL)
                    (LET ((DATABASE (CAR TAIL)))
                         (PROCEED-CASE (CASH-FILE:GET-CASH-FILE NAME (IF (CASH-FILE:CASH-FILE-P DATABASE)
                                                                         DATABASE
                                                                         (LET* ((CASH-FILE (
                                                                                    CASH-FILE:OPEN-CASH-FILE
                                                                                         DATABASE
                                                                                        *WHERE-IS-CASH-SIZE*
                                                                                         ))
                                                                                (HASH-FILE (
                                                                                   CASH-FILE:CASH-FILE-HASH-FILE
                                                                                         CASH-FILE)))
                                                                           ;; install our read function in hash file
                                                                           (SETF (
                                                                            HASH-FILE::HASH-FILE-KEY-READ-FN
                                                                                  HASH-FILE)
                                                                                 #'WHERE-IS-READ-FN)
```

```
                                                                    (SETF (
                                                                HASH-FILE::HASH-FILE-VALUE-READ-FN
                                                                        HASH-FILE)
                                                                       #'WHERE-IS-READ-FN)
                                                                  ;; smash CASH-FILE into
                                                                  ;; *WHERE-IS-CASH-FILES*
                                                                  (SETF (CAR TAIL)
                                                                        CASH-FILE))))
                                        (NIL NIL :REPORT "Delete from the databases known to WHERE-IS?" (
                                                                                          DEL-WHERE-IS-DATABASE
                                                                                                DATABASE)
                                                NIL))))
          *WHERE-IS-CASH-FILES*))
```

```
(DEFUN WHERE-IS-READ-FN (STREAM)
```

;;; the :KEY-READ-FN & :VALUE-READ-FN for WHERE-IS hash files.

```
   (HANDLER-CASE
          ;; use the default read function
          (HASH-FILE::DEFAULT-READ-FN STREAM)
          ;; Quietly handle MISSING-PACKAGE errors by returning the condition.
          ;; This allows us to have files in our database which we havn't loaded.
          (MISSING-PACKAGE (CONDITION)
                 CONDITION)))

(DEFUN ADD-WHERE-IS-DATABASES (&REST PATHNAMES)
   "add each PATHNAME to the databases known to WHERE-IS"
   (MAPCAR #'ADD-WHERE-IS-DATABASE PATHNAMES))

(DEFUN ADD-WHERE-IS-DATABASE (PATHNAME)
   "add PATHNAME to the databases known to WHERE-IS"
   (LET ((NEW-PATHNAME (PATHNAME PATHNAME)))
         ;; first delete & close the old one (if any)
         (DEL-WHERE-IS-DATABASE NEW-PATHNAME)
         ;; now add the new one
         (PUSH NEW-PATHNAME *WHERE-IS-CASH-FILES*)
        NEW-PATHNAME))

(DEFUN DEL-WHERE-IS-DATABASE (DATABASE)
   (LET ((FOUND (FIND-IF #'(LAMBDA (ELEMENT)
                                 (SAME-WHERE-IS-DATABASE DATABASE ELEMENT))
                         *WHERE-IS-CASH-FILES*)))
        (WHEN FOUND
            (SETQ *WHERE-IS-CASH-FILES* (DELETE FOUND *WHERE-IS-CASH-FILES* :TEST 'EQ))
            (IF (CASH-FILE:CASH-FILE-P FOUND)
                (HASH-FILE:CLOSE-HASH-FILE (CASH-FILE:CASH-FILE-HASH-FILE FOUND))
                FOUND))))

(DEFUN SAME-WHERE-IS-DATABASE (X Y)
   (FLET ((COERCE-TO-PATHAME (CASH-FILE-OR-FILE-NAME)
                (PATHNAME (IF (CASH-FILE:CASH-FILE-P CASH-FILE-OR-FILE-NAME)
                              (HASH-FILE::HASH-FILE-STREAM (CASH-FILE:CASH-FILE-HASH-FILE
                                                                CASH-FILE-OR-FILE-NAME))
                              CASH-FILE-OR-FILE-NAME))))
         (LET ((PATHNAME-X (COERCE-TO-PATHAME X))
               (PATHNAME-Y (COERCE-TO-PATHAME Y)))
              ;; do a case & version insensitive comparison
              (AND (EQUALP (PATHNAME-HOST PATHNAME-X)
                           (PATHNAME-HOST PATHNAME-Y))
                   (EQUALP (PATHNAME-DEVICE PATHNAME-X)
                           (PATHNAME-DEVICE PATHNAME-Y))
                   (EQUALP (PATHNAME-DIRECTORY PATHNAME-X)
                           (PATHNAME-DIRECTORY PATHNAME-Y))
                   (EQUALP (PATHNAME-NAME PATHNAME-X)
                           (PATHNAME-NAME PATHNAME-Y))
                   (EQUALP (PATHNAME-TYPE PATHNAME-X)
                           (PATHNAME-TYPE PATHNAME-Y))))))

(DEFUN CLOSE-WHERE-IS-FILES (EVENT)
   (CASE EVENT
       ((NIL IL:BEFORELOGOUT IL:BEFORESYSOUT IL:BEFOREMAKESYS)
          (IL:NLSETQ (MAPLIST #'(LAMBDA (TAIL)
                                     (LET ((CASH-FILE:CASH-FILE (FIRST TAIL)))
                                          (IF (CASH-FILE:CASH-FILE-P CASH-FILE:CASH-FILE)
```

```
                                                    ;; make sure we'll get latest version on re-boot
                                                    (SETF (FIRST TAIL)
                                                          (MAKE-PATHNAME :VERSION :NEWEST :DEFAULTS
                                                                 (HASH-FILE:CLOSE-HASH-FILE (
                                                                        CASH-FILE:CASH-FILE-HASH-FILE
                                                                             CASH-FILE:CASH-FILE))))
)))
                                *WHERE-IS-CASH-FILES*)))))

(IL:ADDTOVAR IL:AROUNDEXITFNS CLOSE-WHERE-IS-FILES)

(DEFVAR *WHERE-IS-CASH-FILES* NIL
   "list of pathnames or CASH-FILEs")

(DEFVAR *WHERE-IS-CASH-SIZE* 100
   "size of the CACHE-FILE cache to use")

;; notice time code

(DEFUN WHERE-IS-NOTICE (DATABASE-FILE &KEY (FILES "*.;")
                                    (NEW NIL)
                                    (DEFINE-TYPES (WHERE-IS-DEFAULT-DEFINE-TYPES))
                                    (HASH-FILE-SIZE *WHERE-IS-HASH-FILE-SIZE*)
                                    (QUIET NIL)
                                    (TEMP-FILE NIL))
   (LET* ((FILE (IF TEMP-FILE
                    (IF NEW
                        TEMP-FILE
                        (IL:COPYFILE DATABASE-FILE TEMP-FILE))
                    DATABASE-FILE))
          (HASH-FILE:HASH-FILE (IF NEW
                                   (HASH-FILE:MAKE-HASH-FILE FILE HASH-FILE-SIZE)
                                   (HASH-FILE:OPEN-HASH-FILE FILE :DIRECTION :IO)))
          (HASH-FILE::*DELETE-OLD-VERSION-ON-REHASH* T))
         (UNWIND-PROTECT
             (DOLIST (PATHNAME (WHERE-IS-FILES FILES))
                 (UNLESS QUIET
                     (FORMAT T ";;; ~A ." (NAMESTRING PATHNAME)))
                 (LET ((NAMESTRING (WHERE-IS-NAMESTRING PATHNAME)))
                      (IF (AND (NOT NEW)
                               (LET ((OLD-WRITE-DATE (WHERE-IS-GET-WRITE-DATE NAMESTRING HASH-FILE:HASH-FILE)))
                                    (AND OLD-WRITE-DATE (= (FILE-WRITE-DATE PATHNAME)
                                                           OLD-WRITE-DATE))))
                          (UNLESS QUIET (FORMAT T " up to date.~%"))
                          (MULTIPLE-VALUE-BIND (FILE-VARS VALUES)
                              (WHERE-IS-READ-COMS PATHNAME)
                            (WHEN FILE-VARS
                                 ;; bind the filevars s.t. IL:INFILECOMS? will find them
                                 (PROGV FILE-VARS VALUES
                                     (UNLESS QUIET (PRINC "."))
                                     (DOLIST (TYPE DEFINE-TYPES)
                                         (LET ((NAMES (IL:INFILECOMS? NIL TYPE (FIRST FILE-VARS))))
                                              (WHEN (CONSP NAMES)
                                                   ;; IL:INFILECOMS? sometimes returns T.
                                                   (DOLIST (NAME NAMES)
                                                        (WHERE-IS-NOTICE-INTERNAL NAME TYPE NAMESTRING
                                                               HASH-FILE:HASH-FILE))))))
                                 (WHERE-IS-SET-WRITE-DATE NAMESTRING PATHNAME HASH-FILE:HASH-FILE)
                                 (UNLESS QUIET
                                     (PRINC ". done.")
                                     (TERPRI)))))))
             (HASH-FILE:CLOSE-HASH-FILE HASH-FILE:HASH-FILE))
         (LET ((PATHNAME (PATHNAME (HASH-FILE::HASH-FILE-STREAM HASH-FILE:HASH-FILE))))
              (COND
                 (TEMP-FILE (UNLESS QUIET
                                (FORMAT T ";;; Renaming ~A ... " (NAMESTRING PATHNAME)))
                            (MULTIPLE-VALUE-BIND (MERGED TRUE-NAME REAL-TRUE-NAME)
                                (RENAME-FILE PATHNAME DATABASE-FILE)
                              (UNLESS QUIET
                                  (FORMAT T "~A~%" (NAMESTRING REAL-TRUE-NAME)))
                              REAL-TRUE-NAME))
                 (T PATHNAME)))))

(DEFUN WHERE-IS-NOTICE-INTERNAL (NAME TYPE FILE-NAME HASH-FILE:HASH-FILE)
    ;; note that NAME is defined as TYPE on FILE-NAME in HASH-FILE
    ;; we keep an ALIST for each name, indexed by type
   (LET* ((ALIST (HASH-FILE:GET-HASH-FILE NAME HASH-FILE:HASH-FILE))
          (OLD-ENTRY (ASSOC TYPE ALIST :TEST 'EQUAL))
```

```
          (OLD-FILES (CDR OLD-ENTRY)))
         (UNLESS (MEMBER FILE-NAME OLD-FILES)
             ;; this optimization helps a lot when re-noticing a file
             (SETF (HASH-FILE:GET-HASH-FILE NAME HASH-FILE:HASH-FILE)
                   (CONS (CONS TYPE (CONS FILE-NAME OLD-FILES))
                         (DELETE OLD-ENTRY ALIST :TEST 'EQ :COUNT 1))))))
```

(DEFUN **WHERE-IS-FILES** (FILES)

```
    ;; expand the FILES argument to WHERE-IS-NOTICE
    ;; allow: non-LIST, file names & file patterns
    (MAPCAN #'(LAMBDA (PATTERN)
                      (LET ((PATHNAME (PROBE-FILE PATTERN)))
                           (IF PATHNAME
                               (LIST PATHNAME)
                               (CASE IL:MAKESYSNAME
                                   (:LYRIC
                                       ;; CL:DIRECTORY is broken in Lyric
                                       (IL:DIRECTORY PATTERN))
                                   (OTHERWISE (DIRECTORY PATTERN))))))
          (IF (LISTP FILES)
              FILES
              (LIST FILES))))
```

(DEFUN **WHERE-IS-DEFAULT-DEFINE-TYPES** ()

```
    (MAPCAN #'(LAMBDA (TYPE)
                      ;; ignore aliases and types on *WHERE-IS-IGNORE-DEFINE-TYPES*
                      (UNLESS (OR (CONSP TYPE)
                                  (MEMBER TYPE *WHERE-IS-IGNORE-DEFINE-TYPES*))
                              (LIST TYPE)))
          IL:FILEPKGTYPES))
```

(DEFUN **WHERE-IS-NAMESTRING** (PATHNAME)

```
    ;; return a namestring for PATHNAME containing only the NAME & TYPE fields
    (NAMESTRING (MAKE-PATHNAME :HOST NIL :NAME (PATHNAME-NAME PATHNAME)
                               :TYPE
                               (IF (EQUAL (PATHNAME-TYPE PATHNAME)
                                          "")
                                   NIL
                                   (PATHNAME-TYPE PATHNAME)))))
```

(DEFUN **WHERE-IS-READ-COMS** (PATHNAME)

;;; returns as first value a list of the filevars on PATHNAME, as second value a list of the values for these filevars.

```
    (IL:RESETLST
        ;; make sure all IL:LOADVARS get undone
        (IL:RESETSAVE (IL:RESETUNDO))
        (DO ((IL:LOAD-VERBOSE-STREAM 'NIL)
             (ALL-FILE-VARS)
             (QUEUE (LIST (IL:FILECOMS (STRING-UPCASE (PATHNAME-NAME PATHNAME))))
                    (COND
                       ((CONSP (IL:NLSETQ (IL:LOADVARS QUEUE PATHNAME NIL)))
                        (MAPCAN #'(LAMBDA (FILE-VAR)
                                          (IF (BOUNDP FILE-VAR)
                                              (LET ((FILE-VARS (IL:INFILECOMS? NIL 'IL:FILEVARS FILE-VAR)))
                                                   (PUSH FILE-VAR ALL-FILE-VARS)
                                                   (WHEN (CONSP FILE-VARS)
                                                         FILE-VARS))
                                              (PROG1 NIL
                                                  (WARN "Couldn't find ~S on ~A." FILE-VAR (NAMESTRING PATHNAME)))
))
                                QUEUE))
                       (T (WARN "Error attempting to LOADVARS ~S from ~A." QUEUE (NAMESTRING PATHNAME))
                          'NIL))))
          ((NULL QUEUE)
           (SETQ ALL-FILE-VARS (NREVERSE ALL-FILE-VARS))
           (VALUES ALL-FILE-VARS (MAPCAR #'SYMBOL-VALUE ALL-FILE-VARS)))
        (DECLARE (SPECIAL IL:LOAD-VERBOSE-STREAM))
        (DOLIST (FILE-VAR QUEUE)
            (IF (MEMBER FILE-VAR ALL-FILE-VARS :TEST 'EQ)
                ;; don't want to load any twice
                (SETF QUEUE (DELETE FILE-VAR QUEUE :TEST 'EQ)))))))
```

(DEFUN **WHERE-IS-SET-WRITE-DATE** (NAMESTRING PATHNAME HASH-FILE:HASH-FILE)

```
    ;; store the write date as a bogus entry on the file
    (WHERE-IS-NOTICE-INTERNAL NAMESTRING 'SI::WRITE-DATE (FILE-WRITE-DATE PATHNAME)
           HASH-FILE:HASH-FILE))

(DEFUN WHERE-IS-GET-WRITE-DATE (NAMESTRING HASH-FILE:HASH-FILE)
    ;; retrieve write date stored for NAMESTRING in HASH-FILE:HASH-FILE
    (CADR (ASSOC 'SI::WRITE-DATE (HASH-FILE:GET-HASH-FILE NAMESTRING HASH-FILE:HASH-FILE))))

(DEFVAR *WHERE-IS-HASH-FILE-SIZE* 10000
   "initial size to create WHERE-IS hash files")

(DEFVAR *WHERE-IS-IGNORE-DEFINE-TYPES* '(IL:FILES IL:EXPRESSIONS IL:FILEVARS IL:ALISTS))

(IL:PUTPROPS IL:WHERE-IS IL:MAKEFILE-ENVIRONMENT (:READTABLE "XCL" :PACKAGE "XCL"))

(IL:PUTPROPS IL:WHERE-IS IL:FILETYPE :COMPILE-FILE)
```

FUNCTION INDEX

ADD-WHERE-IS-DATABASE 2
ADD-WHERE-IS-DATABASES 2
CLOSE-WHERE-IS-FILES 2
DEL-WHERE-IS-DATABASE 2
GET-WHERE-IS-ENTRIES 1
HASH-FILE-TYPES-OF 1
HASH-FILE-WHERE-IS 1
SAME-WHERE-IS-DATABASE 2
WHERE-IS-DEFAULT-DEFINE-TYPES 4
WHERE-IS-FILES 4
WHERE-IS-GET-WRITE-DATE 5
WHERE-IS-NAMESTRING 4
WHERE-IS-NOTICE 3
WHERE-IS-NOTICE-INTERNAL 3
WHERE-IS-READ-COMS 4
WHERE-IS-READ-FN 2
WHERE-IS-SET-WRITE-DATE 4

VARIABLE INDEX

WHERE-IS-CASH-FILES 3
WHERE-IS-CASH-SIZE 3
WHERE-IS-HASH-FILE-SIZE 5
WHERE-IS-IGNORE-DEFINE-TYPES ...5
IL:AROUNDEXITFNS 3

PROPERTY INDEX

IL:WHERE-IS 5