File created: 14-Jun-90 21:15:50 **{DSK}<usr>local>lde>lispcore>internal>library>DO-TEST.;2**

changes to: (IL:VARS IL:DO-TESTCOMS)

previous date: 19-Dec-88 10:03:25 {DSK}<usr>local>lde>lispcore>internal>library>DO-TEST.;1

Read Table: XCL

Package: XCL-USER

Format: XCCS


```
(IL:RPAQQ IL:DO-TESTCOMS
          ((IL:VARIABLES *ANY-ERRORS* *TEST-CLEANUP-FORMS* *TEST-COMPILE* *TEST-MODE* *TEST-BATCH-RESULTS*
                *TEST-FILE-PATTERN* *TEST-FILE-NAME*)
           (IL:P (IMPORT '(DO-TEST-FILE DO-ALL-TESTS DO-TEST DO-TEST-GROUP CL-READFILE EXPECT-ERRORS TEST-DEFUN
                               TEST-DEFMACRO TEST-SETQ *TEST-MODE* *TEST-COMPILE* *TEST-BATCH-RESULTS*
                               *TEST-FILE-PATTERN* *TEST-FILE-NAME*)
                         "XCL-USER")
                 (DEFPACKAGE "XCL-TEST" (:USE "LISP" "XCL")
                        (:IMPORT DO-TEST-FILE DO-ALL-TESTS DO-TEST DO-TEST-GROUP CL-READFILE EXPECT-ERRORS
                               TEST-DEFUN TEST-DEFMACRO TEST-SETQ *TEST-MODE* *TEST-COMPILE*
                               *TEST-BATCH-RESULTS* *TEST-FILE-PATTERN* *TEST-FILE-NAME*)))
           (IL:FUNCTIONS DO-TEST DO-TEST-GROUP MUNG-TEST-FILES PREP-TEST-FILE RUN-COMPILED-TEST-FILES
                  TEST-DEFMACRO TEST-DEFUN TEST-SETQ WITHOUT-BATCH-MODE-ERRORS EXPECT-ERRORS DO-ALL-TESTS
                  CURRENT-FILE-NAME CL-READFILE DO-TEST-FILE DO-TEST-LIST)
          ;; For compiled tests:
           (OPTIMIZERS DO-TEST)
           (IL:COMS
                  ;; Support for saving DO-TESTs onto files.  This defines the TESTS filepkg type, with definers DEFTEST and
                  ;; DEFTESTGROUP.  These expand directly into DO-TEST and DO-TEST-GROUP respectively.
                  (IL:DEFINE-TYPES IL:TESTS)
                  ;; Used to define tests:  These expand exactly into DO-TEST and DO-TEST-GROUP
                  (IL:FUNCTIONS IL:DEFTEST IL:DEFTESTGROUP)
                  ;; This command will run one or more
                  (IL:COMMANDS "RUN")
                  (IL:VARS (IL:*DEFINED-TESTS* (MAKE-HASH-TABLE :SIZE 20 :TEST #'EQUAL)))
                  (IL:GLOBALVARS IL:*DEFINED-TESTS*))
           (IL:COMS
                  ;;  ster definition FOR VERIFIED TESTS.
                  (IL:DEFINE-TYPES VERIFIED-TESTS)
                  (IL:FUNCTIONS TEST-EQUAL)
                  (IL:FUNCTIONS DEFINE-VERIFIED-TEST MAKE-TEST-DEFUN)
                  (IL:FUNCTIONS VERIFIED-TEST-TO-DO-TEST GET-FORMS COMMENT-P)
                  (IL:FUNCTIONS TYPE-NUMBER MAKE-TEST-DEFUN)
                  (IL:COMMANDS "COPY-TEST" "E-TEST"))
           (IL:PROP (IL:FILETYPE)
                  DO-TEST)))

(DEFVAR *ANY-ERRORS* NIL)

(DEFVAR *TEST-CLEANUP-FORMS* NIL)

(DEFVAR *TEST-COMPILE* NIL)

(DEFVAR *TEST-MODE* :BATCH)

(DEFVAR *TEST-BATCH-RESULTS* "{eris}<Test>Language>Auto>test-results")

(DEFVAR *TEST-FILE-PATTERN* '("{ERIS}<LISPCORE>CML>TEST>*.TEST;" "{ERIS}<LISPCORE>CML>TEST>*.X;"
                                    "{ERIS}<LISPCORE>PATCHES>TESTS>*.TEST;" "{ERIS}<LISPCORE>TEST>*.TEST;"))

(DEFVAR *TEST-FILE-NAME* "unknown")

(IMPORT '(DO-TEST-FILE DO-ALL-TESTS DO-TEST DO-TEST-GROUP CL-READFILE EXPECT-ERRORS TEST-DEFUN TEST-DEFMACRO
             TEST-SETQ *TEST-MODE* *TEST-COMPILE* *TEST-BATCH-RESULTS* *TEST-FILE-PATTERN* *TEST-FILE-NAME*)
       "XCL-USER")

(DEFPACKAGE "XCL-TEST" (:USE "LISP" "XCL")
       (:IMPORT DO-TEST-FILE DO-ALL-TESTS DO-TEST DO-TEST-GROUP CL-READFILE EXPECT-ERRORS TEST-DEFUN
              TEST-DEFMACRO TEST-SETQ *TEST-MODE* *TEST-COMPILE* *TEST-BATCH-RESULTS* *TEST-FILE-PATTERN*
              *TEST-FILE-NAME*))
```

```
(DEFMACRO DO-TEST (IL:NAME-AND-OPTIONS &BODY IL:BODY)
   ;; This is the MACRO for DO-TEST; it does NOT run compiled tests.
   ;; !!!*****If you change this, change the optimizer, too*****!!!
   (LET ((IL:NAME NIL)
         (IL:OPTIONS NIL))
        (COND
           ((CONSP IL:NAME-AND-OPTIONS)
            (SETQ IL:NAME (CAR IL:NAME-AND-OPTIONS))
            (SETQ IL:OPTIONS (CDR IL:NAME-AND-OPTIONS)))
           (T (SETQ IL:NAME IL:NAME-AND-OPTIONS)))
        (IF (OR (EQ *TEST-MODE* :INTERACTIVE)
                (EQ *TEST-MODE* :BATCH-VERBOSE))
            (FORMAT *ERROR-OUTPUT* "Testing... ~S~%" IL:NAME))
        (COND
           ((IL:FMEMB :COMPILED IL:OPTIONS)
            `(FORMAT *ERROR-OUTPUT* "Compiled-only test \"~A\" in file \"~A\" not run.~%" ',IL:NAME (
                                                                                    CURRENT-FILE-NAME
                                                                                          )))
           (T `(NOT (WHEN (NULL (WITHOUT-BATCH-MODE-ERRORS ,@IL:BODY))
                      (FORMAT *ERROR-OUTPUT* "Test \"~A\" failed in file \"~A\"~%" ',IL:NAME (
                                                                                    CURRENT-FILE-NAME
                                                                                     ))
                      (IL:SETQ *ANY-ERRORS* T)))))))


(DEFMACRO DO-TEST-GROUP (IL:NAME-AND-OPTIONS &BODY IL:BODY)
   (LET ((IL:NAME NIL)
         (IL:OPTIONS NIL))
        (COND
           ((CONSP IL:NAME-AND-OPTIONS)
            (SETQ IL:NAME (CAR IL:NAME-AND-OPTIONS))
            (SETQ IL:OPTIONS (CDR IL:NAME-AND-OPTIONS)))
           (T (SETQ IL:NAME IL:NAME-AND-OPTIONS)))
        ;; Hack: find :BEFORE and :AFTER clauses in the body and move them out
        (LOOP (IF (AND (SYMBOLP (CAR IL:BODY))
                       (OR (EQ (CAR IL:BODY)
                               :BEFORE)
                           (EQ (CAR IL:BODY)
                               :AFTER)))
                  (PROGN (IL:SETQ IL:OPTIONS (IL:APPEND IL:OPTIONS (LIST (CAR IL:BODY)
                                                                         (CADR IL:BODY))))
                         (IL:SETQ IL:BODY (CDDR IL:BODY)))
                  (RETURN NIL)))
        `(LET ((*TEST-CLEANUP-FORMS* NIL))
              ,@(AND (BOUNDP '*COMPILING-TEST-FILES*)
                     `((EVAL-WHEN (COMPILE)
                                  (SETQ *TEST-CLEANUP-FORMS* NIL)
                                 ,(IGNORE-ERRORS (GETF IL:OPTIONS :BEFORE)))))
              (BLOCK ,IL:NAME
                  ,(IF (OR (EQ *TEST-MODE* :INTERACTIVE)
                           (EQ *TEST-MODE* :BATCH-VERBOSE))
                       (FORMAT *ERROR-OUTPUT* "Testing... ~S~%" IL:NAME))
                  ,(LET ((IL:BEFORE (IGNORE-ERRORS (GETF IL:OPTIONS :BEFORE))))
                        (IF IL:BEFORE
                            `(WHEN (NULL (WITHOUT-BATCH-MODE-ERRORS ,IL:BEFORE T))
                                 (FORMAT *ERROR-OUTPUT* ":BEFORE forms for test \"~A\" in file ~S failed."
                                        ',IL:NAME
                                        (CURRENT-FILE-NAME))
                                 (IL:SETQ *ANY-ERRORS* T)
                                 (RETURN-FROM ,IL:NAME))))
                  ,@(IL:for IL:B IL:in IL:BODY IL:join (COND
                                                         ((AND (CONSP IL:B)
                                                               (EQ (CAR IL:B)
                                                                   'DO-TEST))
                                                          ;; DO-TESTs work fine.
                                                          (LIST IL:B))
                                                         ((AND (CONSP IL:B)
                                                               (EQ (CAR IL:B)
                                                                   'IL:DEFTEST))
                                                          ;; eplace all the DEFTESTs with DO-TESTs
                                                          (LIST (CONS 'DO-TEST (CDR IL:B))))
                                                         (T (FORMAT *ERROR-OUTPUT* "Non DO-TEST form in ~S in
                                                                    ~S~%~S~%" IL:NAME (CURRENT-FILE-NAME)
                                                                    IL:B))))
                  ,(LET ((IL:AFTER (IGNORE-ERRORS (GETF IL:OPTIONS :AFTER))))
                        (IF IL:AFTER
                            `(WHEN (NULL (WITHOUT-BATCH-MODE-ERRORS ,IL:AFTER T))
                                 (FORMAT *ERROR-OUTPUT* ":AFTER forms for test \"~A\" in file ~S failed."
                                        ',IL:NAME
                                        (CURRENT-FILE-NAME))
                                 (SETQ *ANY-ERRORS* T)))))
              (EVAL (CONS 'PROGN *TEST-CLEANUP-FORMS*))
              ,@(AND (BOUNDP '*COMPILING-TEST-FILES*)
```

```
                    `((EVAL-WHEN (COMPILE)
                            (EVAL (CONS 'PROGN *TEST-CLEANUP-FORMS*)))))
              NIL)))

(DEFUN MUNG-TEST-FILES (PATTERN &KEY (COMPILER 'COMPILE-FILE)
                                (STARTINGLIST))
   (LET ((*COMPILING-TEST-FILES* T))
        (DECLARE (SPECIAL *COMPILING-TEST-FILES*))
        (DOLIST (PN (OR STARTINGLIST (DIRECTORY PATTERN)))
            (LET* ((LOCALFILE (PREP-TEST-FILE PN))
                   (COMPILED-FILE (IGNORE-ERRORS (FUNCALL COMPILER LOCALFILE))))
                  (IF COMPILED-FILE
                      (PROGN (IL:COPYFILE COMPILED-FILE (NAMESTRING (MAKE-PATHNAME :TYPE (PATHNAME-TYPE
                                                                                          COMPILED-FILE)
                                                                                   :DEFAULTS PN)))
                             (DELETE-FILE COMPILED-FILE))
                      (FORMAT *ERROR-OUTPUT* "Couldn't compile ~a~%" PN))))))

(DEFUN PREP-TEST-FILE (FILE)
   (LET ((OUTFILE (IL:OPENFILE "{core}hack.;1" 'IL:OUTPUT)))
        (FORMAT OUTFILE ";; This is a COMMON LISP FILE, DAMMIT!
               (setq XCL-USER::*TEST-FILE-NAME* ~S)
               (in-package \"XCL-TEST\")
              " (PATHNAME-NAME FILE))
        (IL:COPYFILE FILE OUTFILE)
        (CLOSE OUTFILE)
        "{core}hack.;1"))

(DEFUN RUN-COMPILED-TEST-FILES (PATTERN)
   ;; I'd like to use IL:DIRECTORY and generate the list incrementally, but IL:DIRECTORY returns upcased filenames, which Unix dislikes... *sigh*...
   (DOLIST (PN (DIRECTORY PATTERN))
       (LET ((*TEST-FILE-NAME* (PATHNAME-NAME PN)))
            (LOAD PN))))

(DEFMACRO TEST-DEFMACRO (IL:NAME &REST IL:STUFF)
   `(PROGN (IF (FBOUNDP ',IL:NAME)
               (IF (MACRO-FUNCTION ',IL:NAME)
                   (PUSH (LIST 'SETF (LIST 'SYMBOL-FUNCTION (LIST 'MACRO-FUNCTION ''',IL:NAME))
                                 (LIST 'QUOTE (SYMBOL-FUNCTION (MACRO-FUNCTION ',IL:NAME))))
                         *TEST-CLEANUP-FORMS*)
                   (ERROR "Please don't redefine ~A in a test form" ',IL:NAME))
               (PUSH (LIST 'REMPROP ''',IL:NAME ''IL:MACRO-FN)
                     *TEST-CLEANUP-FORMS*))
           (DEFMACRO ,IL:NAME ,@IL:STUFF)))

(DEFMACRO TEST-DEFUN (IL:NAME &REST IL:STUFF)
   `(PROGN (IF (FBOUNDP ',IL:NAME)
               (IF (OR (MACRO-FUNCTION ',IL:NAME)
                       (SPECIAL-FORM-P ',IL:NAME))
                   (ERROR "Please don't redefine ~A in a test form" ',IL:NAME)
                   (PUSH (LIST 'SETF (LIST 'SYMBOL-FUNCTION ''',IL:NAME)
                                 (LIST 'QUOTE (SYMBOL-FUNCTION ',IL:NAME)))
                         *TEST-CLEANUP-FORMS*))
               (PUSH (LIST 'FMAKUNBOUND ''',IL:NAME)
                     *TEST-CLEANUP-FORMS*))
           (IL:EVAL '(DEFUN ,IL:NAME ,@IL:STUFF))))

(DEFMACRO TEST-SETQ (&REST STUFF)
   (LET (UNBINDLIST)
        (DO ((X STUFF (CDDR X)))
            ((NULL X))
          (PUSH `(IF (BOUNDP ',(CAR X))
                     (PUSH (LIST 'SETQ ',(CAR X)
                                 (LIST 'QUOTE (SYMBOL-VALUE ',(CAR X))))
                           *TEST-CLEANUP-FORMS*)
                     (PUSH (LIST 'MAKUNBOUND ''',(CAR X))
                           *TEST-CLEANUP-FORMS*))
                UNBINDLIST))
        `(PROGN ,@UNBINDLIST (SETQ ,@STUFF))))

(DEFMACRO WITHOUT-BATCH-MODE-ERRORS (&BODY IL:BODY)
   (COND
      ((EQ *TEST-MODE* :INTERACTIVE)
       `(PROGN ,@IL:BODY))
      (T `(IGNORE-ERRORS ,@IL:BODY))))

(DEFMACRO EXPECT-ERRORS (IL:ERROR-TYPES &REST IL:FORMS)
   `(HANDLER-CASE (PROGN ,@IL:FORMS NIL)
```

```
           (,(IF (LISTP IL:ERROR-TYPES)
                 `(OR ,@IL:ERROR-TYPES)
                 IL:ERROR-TYPES)
            (CONDITION)
            (VALUES T CONDITION))))


(DEFUN DO-ALL-TESTS (&KEY (RESULTS *TEST-BATCH-RESULTS*)
                          (PATTERNS (IF (CONSP *TEST-FILE-PATTERN*)
                                        *TEST-FILE-PATTERN*
                                        (LIST *TEST-FILE-PATTERN*)))
                          (SYSOUT-TYPE NIL)
                          (RESUME NIL))
   (LET ((IL:NO-PROBLEMS T)
         (*DEFAULT-PATHNAME-DEFAULTS* (PATHNAME "{ERIS}<LISPCORE>CML>TEST>"))
         (*ERROR-OUTPUT* (IF (EQ RESULTS T)
                             *ERROR-OUTPUT*
                             (OPEN RESULTS :DIRECTION :OUTPUT :IF-EXISTS (IF RESUME
                                                                             :APPEND
                                                                             :NEW-VERSION)))))
        (UNWIND-PROTECT
            (PROGN (IF (NOT RESUME)
                       (PROGN (FORMAT *ERROR-OUTPUT* ";;; Test results for sysout of ~A~%" IL:MAKESYSDATE)
                              (IF SYSOUT-TYPE (FORMAT *ERROR-OUTPUT* ";;; Sysout type is ~A~%" SYSOUT-TYPE))
                              (IF *TEST-COMPILE* (FORMAT *ERROR-OUTPUT* ";;; Tests are being compiled~%"))
                              (FORMAT *ERROR-OUTPUT* ";;; Tests run on ~A~%" (IL:DATE))
                              (FORMAT *ERROR-OUTPUT* ";;; Running tests from ~A~2%" PATTERNS)
                              (SETQ *ALL-FILES-REMAINING* (IL:FOR DP IL:IN PATTERNS IL:JOIN (IL:DIRECTORY DP))))
                       (FORMAT *ERROR-OUTPUT* ";;;Resuming after dying on file ~S~%" (POP *ALL-FILES-REMAINING*)
                               ))
                   (IL:|while| *ALL-FILES-REMAINING* IL:|do| (IL:SETQ IL:NO-PROBLEMS (AND (DO-TEST-FILE (CAR
                                                                                                          *ALL-FILES-REMAINING*
                                                                                                          ))
                                                                                         IL:NO-PROBLEMS))
                                                             (IL:|pop| *ALL-FILES-REMAINING*))
                   (FORMAT *ERROR-OUTPUT* "(END-OF-TESTS)"))
            (UNLESS (EQ RESULTS T)
                    (CLOSE *ERROR-OUTPUT*)))
          IL:NO-PROBLEMS))

(DEFMACRO CURRENT-FILE-NAME ()
   `',*TEST-FILE-NAME*)


(DEFUN CL-READFILE (IL:TEST-FILE &OPTIONAL (*READTABLE* IL:CMLRDTBL)
                                 (IL:ENDTOKEN "STOP"))
   (LET ((TRUE-NAME (PROBE-FILE IL:TEST-FILE))
         IL:FILE-CONTENTS)
        (IF TRUE-NAME
            (COND
               ((IL:LISPSOURCEFILEP TRUE-NAME)
                ;; IL-format (managed) file.  Read it using IL:READFILE and drop the DEFINE-FILE-INFO form on the front (if there is one), so that
                ;; it all reads with the right package.
                (HANDLER-BIND ((WARNING #'(LAMBDA (THE-WARNING)
                                                  (FORMAT *ERROR-OUTPUT* "~&While reading ~S: ~A~%" (
                                                                                                    CURRENT-FILE-NAME
                                                                                                    )
                                                          THE-WARNING)
                                                  (MUFFLE-WARNING))))
                               (REMOVE-IF #'(IL:LAMBDA (IL:FORM)
                                                       (AND (IL:LISTP IL:FORM)
                                                            (IL:FMEMB (CAR IL:FORM)
                                                                      '(IL:DEFINE-FILE-INFO IL:DECLARE\:))))
                                          (IL:READFILE TRUE-NAME))))
               (T ;; CL format file.  Have to read it carefully.
                  (LET (IL:TEM (*PACKAGE* (FIND-PACKAGE 'XCL-TEST))
                               (*FEATURES* (CONS :NO-STACK-OVERFLOW *FEATURES*)))
                       (SETQ *TEST-FILE-NAME* (NAMESTRING TRUE-NAME))
                       (WITH-COLLECTION (WITH-OPEN-STREAM
                                         (IL:TEST-FILE (IL:OPENTEXTSTREAM (IL:MKATOM IL:TEST-FILE)))
                                         (HANDLER-BIND ((WARNING #'(LAMBDA (THE-WARNING)
                                                                           (FORMAT *ERROR-OUTPUT* "~&While
                                                                                   reading ~S: ~A~%" (
                                                                                                     CURRENT-FILE-NAME
                                                                                                     )
                                                                                   THE-WARNING)
                                                                           (MUFFLE-WARNING))))
                                                        (LOOP (WHEN (OR (NULL (IGNORE-ERRORS (SETQ IL:TEM (READ
                                                                                                           IL:TEST-FILE
                                                                                                           ))))
                                                                        (AND (SYMBOLP IL:TEM)
                                                                             (STRING= IL:TEM IL:ENDTOKEN)))
                                                                    (RETURN))
                                                              (COLLECT IL:TEM)))))))))))
```

```
            (PROGN (FORMAT *ERROR-OUTPUT* "~%Couldn't find file ~A~%" IL:TEST-FILE)
                   NIL))))

(DEFUN DO-TEST-FILE (IL:FILENAME)
   (LET* ((*PACKAGE* (FIND-PACKAGE 'XCL-TEST))
          (*TEST-FILE-NAME* NIL)
          (IL:TEST-FORMS (CL-READFILE IL:FILENAME IL:CMLRDTBL))
          (*ANY-ERRORS* NIL))
         (DO-TEST-LIST IL:TEST-FORMS)
         (IL:|if| *ANY-ERRORS*
             IL:|then| (TERPRI *ERROR-OUTPUT*))
         (NOT *ANY-ERRORS*)))

(DEFUN DO-TEST-LIST (TEST-FORMS &OPTIONAL OPTIONS NAME)
    ;; Runs thru a list of test froms from a file, executing them one-by-one.  If it finds non-test-looking forms, it warns you, but evaluates the forms
    ;; anyhow.
    (LET
     ((IL:DFNFLG NIL))
     (DECLARE (SPECIAL IL:DFNFLG))
     (IL:|if| (NULL TEST-FORMS)
         IL:|then| (FORMAT *ERROR-OUTPUT* "~%(Trouble reading ~A)~%" (CURRENT-FILE-NAME))
                (SETQ *ANY-ERRORS* T)
       IL:|else|
       (IL:|for| FORM IL:|in| TEST-FORMS
          IL:|do| (IL:BLOCK 0)
                  (IF (AND (CONSP FORM)
                           (IL:FMEMB (CAR FORM)
                                     '(DO-TEST DO-TEST-GROUP DEFTEST DEFTESTGROUP IL:DEFTEST IL:DEFTESTGROUP)))
                      (HANDLER-BIND ((WARNING #'(LAMBDA (THE-WARNING)
                                                  (FLET ((TEST-NAME NIL (LET ((X (SECOND FORM)))
                                                                          (IF (CONSP X)
                                                                              (FIRST X)
                                                                             X))))
                                                    (FORMAT *ERROR-OUTPUT* "~&Warning in test ~S in file
                                                           ~S:~&  ~A~%" (TEST-NAME)
                                                           (CURRENT-FILE-NAME)
                                                           THE-WARNING)
                                                    (MUFFLE-WARNING)))))
                                    (IF *TEST-COMPILE*
                                        (BLOCK COMPILER-PUNT
                                            (LET ((COMPILED-FORM (IF (EQ *TEST-MODE* :INTERACTIVE)
                                                                     (COMPILE NIL `(LAMBDA NIL ,FORM))
                                                                     (IGNORE-ERRORS (COMPILE NIL
                                                                                             `(LAMBDA NIL ,FORM))))))
                                                 (IF (NULL (COMPILED-FUNCTION-P COMPILED-FORM))
                                                     (LET ((*PRINT-LEVEL* 3)
                                                           (*PRINT-LENGTH* 3))
                                                          (FORMAT *ERROR-OUTPUT* "Compilation of this form in file ~S
                                                                 failed:~%   ~S~%" (CURRENT-FILE-NAME)
                                                                 FORM)
                                                          (RETURN-FROM COMPILER-PUNT))
                                                     (IF (NULL (IF (EQ *TEST-MODE* :INTERACTIVE)
                                                                   (PROGN (FUNCALL COMPILED-FORM)
                                                                          T)
                                                                   (IGNORE-ERRORS (PROGN (FUNCALL COMPILED-FORM)
                                                                                         T))))
                                                         (LET ((*PRINT-LEVEL* 3)
                                                               (*PRINT-LENGTH* 3))
                                                              (FORMAT *ERROR-OUTPUT* "Compiled code failed for this form
                                                                     in file ~S :~%~S~%" (CURRENT-FILE-NAME)
                                                                     FORM))))))
                                        (EVAL FORM)))
                      (PROGN (UNLESS (MEMBER (CAR FORM)
                                             '(IL:DEFINE-FILE-INFO IL:FILECREATED IL:PRETTYCOMPRINT IL:FILESLOAD
                                                    IL:RPAQQ IL:RPAQ IL:RPAQ? IL:*))
                                     (LET ((*PRINT-LEVEL* 3)
                                           (*PRINT-LENGTH* 3))
                                          (FORMAT *ERROR-OUTPUT* "Non DO-TEST form at top level in ~S~%~S~%" (
                                                                                                    CURRENT-FILE-NAME
                                                                                                    )
                                                  FORM)))
                             ;; Evaluate the form anyway.
                             (EVAL FORM))))))

;; For compiled tests:

(DEFOPTIMIZER DO-TEST (IL:NAME-AND-OPTIONS &BODY IL:BODY)
                      ;; This is the version of DO-TEST for compiling tests.  It DOESN'T check to see if this is a compiled-only test, but DOES
                      ;; 9or will) check to see if it's an interpreted-only test.
                      ;; !!!!!*****If you change the semantics of this, change the MACRO too*****!!!!!
```

```
                        (LET ((IL:NAME NIL)
                              (IL:OPTIONS NIL))
                             (COND
                                ((CONSP IL:NAME-AND-OPTIONS)
                                 (SETQ IL:NAME (CAR IL:NAME-AND-OPTIONS))
                                 (SETQ IL:OPTIONS (CDR IL:NAME-AND-OPTIONS)))
                                (T (SETQ IL:NAME IL:NAME-AND-OPTIONS)))
                             (IF (OR (EQ *TEST-MODE* :INTERACTIVE)
                                     (EQ *TEST-MODE* :BATCH-VERBOSE))
                                 (FORMAT *ERROR-OUTPUT* "Testing... ~S~%" IL:NAME))
                             (COND
                                ((OR (IL:FMEMB :INTERPRET IL:OPTIONS)
                                     (IL:FMEMB :INTERPRET-ONLY IL:OPTIONS)
                                     (IL:FMEMB :INTERPRETED-ONLY IL:OPTIONS)
                                     (IL:FMEMB :INTERPRETED IL:OPTIONS))
                                 '(FORMAT *ERROR-OUTPUT* "Interpreted-only test \"~A\" in file \"~A\" not
                                          run.~%" ',IL:NAME (CURRENT-FILE-NAME)))
                                (T `(NOT (WHEN (NULL (WITHOUT-BATCH-MODE-ERRORS ,@IL:BODY))
                                               (FORMAT *ERROR-OUTPUT* "Test \"~A\" failed in file \"~A\"~%"
                                                       ',IL:NAME
                                                       (CURRENT-FILE-NAME))
                                               (IL:SETQ *ANY-ERRORS* T)))))))
```

;; Support for saving DO-TESTs onto files. This defines the TESTS filepkg type, with definers DEFTEST and DEFTESTGROUP. These expand directly
;; into DO-TEST and DO-TEST-GROUP respectively.

```
(DEF-DEFINE-TYPE IL:TESTS "Tests for automated regression testing")
```

;; Used to define tests: These expand exactly into DO-TEST and DO-TEST-GROUP

```
(DEFDEFINER (IL:DEFTEST (:NAME (IL:LAMBDA (IL:BODY)
                                  (COND
                                     ((IL:LISTP (CADR IL:BODY))
                                      (CAADR IL:BODY))
                                     (T (CADR IL:BODY))))))
    IL:TESTS (IL:NAME-AND-OPTIONS &BODY IL:BODY)
   (LET ((IL:NAME (COND
                     ((IL:LISTP IL:NAME-AND-OPTIONS)
                      (CAR IL:NAME-AND-OPTIONS))
                     (T IL:NAME-AND-OPTIONS)))
         (IL:TEST-FORM `(DO-TEST ,IL:NAME-AND-OPTIONS ,@IL:BODY)))
        (SETF (GETHASH IL:NAME IL:*DEFINED-TESTS*)
              IL:TEST-FORM)
        `(DO-TEST ,IL:NAME-AND-OPTIONS ,@IL:BODY)))

(DEFDEFINER (IL:DEFTESTGROUP (:NAME (IL:LAMBDA (IL:BODY)
                                       (COND
                                          ((IL:LISTP (CADR IL:BODY))
                                           (CAADR IL:BODY))
                                          (T (CADR IL:BODY))))))
    IL:TESTS (IL:NAME-AND-OPTIONS &BODY IL:BODY)
   (LET ((IL:NAME (COND
                     ((IL:LISTP IL:NAME-AND-OPTIONS (CAR IL:NAME-AND-OPTIONS)))
                     (T IL:NAME-AND-OPTIONS)))
         (IL:TEST-FORM `(DO-TEST-GROUP ,IL:NAME-AND-OPTIONS ,@IL:BODY)))
        (SETF (GETHASH IL:NAME IL:*DEFINED-TESTS*)
              IL:TEST-FORM)
        `(DO-TEST-GROUP ,IL:NAME-AND-OPTIONS ,@IL:BODY)))
```

;; This command will run one or more

```
(DEFCOMMAND "RUN" (&REST IL:TESTS) (IL:FOR IL:TEST IL:IN IL:TESTS IL:DO (IL:EVAL (GETHASH IL:TEST
                                                                                   IL:*DEFINED-TESTS*))))

(IL:RPAQ IL:*DEFINED-TESTS* (MAKE-HASH-TABLE :SIZE 20 :TEST #'EQUAL))

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

(IL:GLOBALVARS IL:*DEFINED-TESTS*)
)
```

;; ster definition FOR VERIFIED TESTS.

```
(DEF-DEFINE-TYPE VERIFIED-TESTS "verified regression test")

(DEFUN TEST-EQUAL (X Y)
   (EQUAL X Y))

(DEFDEFINER DEFINE-VERIFIED-TEST VERIFIED-TESTS (NAME MESSAGE &BODY BODY)
    ;; Assumes the body is a form that returns a value or a list of values (comparable by equal) that may be computed at definition/compile time. NAME
    ;; is a symbol and MESSAGE is a string to printed at success/failure
```

```
   (LET ((VALUES (COMPILE-FORM `(PROGN ,@BODY))))
        `(EVAL-WHEN (LOAD)
                (FORMAT *ERROR-OUTPUT* "~&Test: ~a, " ,MESSAGE)
                (IF (TEST-EQUAL ',VALUES (PROGN ,@BODY))
                    (FORMAT *ERROR-OUTPUT* "succeeded.~%")
                    (FORMAT *ERROR-OUTPUT* "failed. ***********~%")))))

(DEFMACRO MAKE-TEST-DEFUN (TEST-NAME)
   `(DEFUN ,TEST-NAME ()
      ,@(NTHCDR 3 (IL:GETDEF TEST-NAME 'VERIFIED-TESTS))))

(DEFUN VERIFIED-TEST-TO-DO-TEST (FILENAME PATHNAME &OPTIONAL (LINELENGTH 60))
   (LET* ((ROOT-NAME (INTERN (STRING FILENAME)
                             (FIND-PACKAGE "INTERLISP")))
          (MAKEFILE-ENVIRONMENT (GET ROOT-NAME 'IL:MAKEFILE-ENVIRONMENT)))
         (LET ((*PACKAGE* (FIND-PACKAGE (OR (SECOND (MEMBER :PACKAGE MAKEFILE-ENVIRONMENT :TEST #'EQ))
                                            "INTERLISP")))
               (*READTABLE* (IL:FIND-READTABLE (OR (SECOND (MEMBER :READTABLE MAKEFILE-ENVIRONMENT :TEST
                                                                   #'EQ))
                                                   "INTERLISP")))
               (*PRINT-BASE* (OR (SECOND (MEMBER :BASE MAKEFILE-ENVIRONMENT :TEST #'EQ))
                                 10))
               (*PRINT-CASE* :DOWNCASE)
               (*PRINT-ARRAY* T)
               (*PRINT-LEVEL* NIL)
               (*PRINT-LENGTH* NIL)
               (*PRINT-STRUCTURE* T)
               ;; Interlisp gorp that controls pretty printing
               (IL:*PRINT-SEMICOLON-COMMENTS* T)
               (IL:FONTCHANGEFLG NIL)
               (IL:\#RPARS NIL)
               (IL:**COMMENT**FLG NIL))
              (DECLARE (GLOBAL IL:FILELINELENGTH IL:PRETTYFLG))
              (DECLARE (SPECIAL IL:FONTCHANGEFLG IL:\#RPARS IL:**COMMENT**FLG IL:*PRINT-SEMICOLON-COMMENTS*))
              (WITH-OPEN-FILE (STREAM (MAKE-PATHNAME :TYPE "TEST" :VERSION :NEWEST :DEFAULTS PATHNAME)
                                      :DIRECTION :OUTPUT)
                     (IL:RESETVARS
                      ;; Interlisp gorp that controls pretty printing
                                   ((IL:FILELINELENGTH LINELENGTH)
                                    (IL:PRETTYFLG T))
                      ;; Identifier
                                   (FORMAT STREAM "~&;;; File converted on ~A from source ~A" (IL:DATE)
                                          ROOT-NAME)
                                   (LET ((DATES (GET ROOT-NAME 'IL:FILEDATES)))
                                        (WHEN DATES
                                            (FORMAT STREAM "~&;;; Original source ~A created ~A" (CDAR DATES)
                                                   (CAAR DATES))))
                                   (TERPRI STREAM)
                                   (TERPRI STREAM)
                      ;; Copyright notice
                                   (LET ((OWNER (GET ROOT-NAME 'IL:COPYRIGHT)))
                                        (WHEN (AND OWNER (CONSP OWNER))
                                            (FORMAT STREAM ";;; Copyright (c) ")
                                            (DO ((TAIL (CDR OWNER)
                                                       (CDR TAIL)))
                                                ((NULL TAIL))
                                              (FORMAT STREAM "~4d" (CAR TAIL))
                                              (IF (CDR TAIL)
                                                  (PRINC ", " STREAM)))
                                            (FORMAT STREAM " by ~a~%" (CAR OWNER))))
                                   (TERPRI STREAM)
                                   (DOLIST (COM (SYMBOL-VALUE (IL:FILECOMS ROOT-NAME)))
                                       (DOLIST (FORM (GET-FORMS COM))
                                           (PPRINT FORM STREAM)
                                           (TERPRI STREAM)
                                           (IL:BLOCK))))
                     (NAMESTRING STREAM)))))

(DEFUN GET-FORMS (COMMAND)
   (LET
    ((UNSUPPORTED-TYPES '(IL:FNS IL:SPECVARS IL:GLOBALVARS IL:LOCALVARS IL:INITVARS IL:ALISTS IL:DEFS
                          IL:INITRECORDS IL:LISPXMACROS IL:MACROS IL:PROPS IL:RECORDS IL:SYSRECORDS
                          IL:USERMACROS IL:VARS IL:CONSTANTS EXPORT IL:RESOURCES IL:INITRESOURCES
                          IL:GLOBALRESOURCES IL:I.S.OPRS IL:HORRIBLEVARS IL:UGLYVARS IL:BITMAPS IL:CURSORS
                          IL:ADVICE IL:ADVISE IL:COURIERPROGRAMS IL:TEMPLATES IL:PROP IL:FILES
                          IL:DECLARE\:))
     (FILEPKGTYPE (CAR COMMAND)))
    (IF (MEMBER FILEPKGTYPE UNSUPPORTED-TYPES :TEST #'EQ)
        (PROGN (WARN "Filepkg type ~s not supported: ~s" FILEPKGTYPE COMMAND)
```

```
            NIL)
        (CASE FILEPKGTYPE
            (IL:P (CDR COMMAND))
            (IL:COMS
               ;; Recurse
               (MAPCAN #'(LAMBDA (X)
                                 (GET-FORMS X))
                       (CDR COMMAND)))
            ((EVAL-WHEN IL:EVAL-WHEN) `((EVAL-WHEN ,(MAPCAR #'(LAMBDA (SYM)
                                                                 (INTERN (STRING SYM)
                                                                         (FIND-PACKAGE "LISP")))
                                                            (SECOND COMMAND))
                                          ,@(GET-FORMS (THIRD COMMAND)))))
            ((IL:*)
               ;; Comment
               (LIST COMMAND))
            (T ;; Should the filepkgtype of a definer
               (LET
                ((IGNORED-DEFINERS '(FILE-ENVIRONMENTS IL:DEFINE-TYPES OPTIMIZERS IL:SEDIT-FORMATS
                                     ADVISED-FUNCTIONS IL:COMMANDS IL:SPECIAL-FORMS PROFILES
                                     XCL::WALKER-TEMPLATES))
                 (DEFINER-TYPE (IL:GETFILEPKGTYPE FILEPKGTYPE 'IL:COMMANDS T)))
                (IF (MEMBER DEFINER-TYPE IGNORED-DEFINERS :TEST #'EQ)
                    (UNLESS (EQ DEFINER-TYPE 'FILE-ENVIRONMENTS)
                        (PROGN (WARN "Ignoring definer coms: ~s" COMMAND)
                               NIL))
                    (LET*
                     ((GET-DEF-METHOD (AND DEFINER-TYPE (GET DEFINER-TYPE :DEFINED-BY)
                                           (GET DEFINER-TYPE 'IL:GETDEF)))
                      (DEFS (AND GET-DEF-METHOD (MAPCAR #'(LAMBDA (NAME)
                                                            (IF (COMMENT-P NAME)
                                                                NAME
                                                                (FUNCALL GET-DEF-METHOD NAME DEFINER-TYPE))
                                                            )
                                                        (CDR COMMAND)))))
                     (CASE DEFINER-TYPE
                         (VERIFIED-TESTS
                            (SETQ
                             DEFS
                             (MAPCAR
                              #'(LAMBDA
                                  (DEF)
                                  (DESTRUCTURING-BIND
                                   (TAG NAME MESSAGE &BODY BODY)
                                   DEF
                                   (LET ((VALUES (COMPILE-FORM (REMOVE-COMMENTS `(PROGN ,@BODY)))))
                                        `(DO-TEST ,MESSAGE (EQUAL ',VALUES
                                                                  ,@(IF (EQ 1 (LENGTH BODY))
                                                                        BODY
                                                                        `((PROGN ,@BODY))))))))))
                              DEFS))))
                     (OR DEFS (PROGN (WARN "Can't parse: ~s" COMMAND)
                                     NIL)))))))))

(DEFUN COMMENT-P (FORM)
   (AND (CONSP FORM)
        (EQ (CAR FORM)
            'IL:*)
        (CONSP (CDR FORM))
        (MEMBER (CADR FORM)
                '(IL:\; IL:|;;| IL:|;;;|)
                :TEST
                #'EQ)
        T))

(DEFUN TYPE-NUMBER (TYPE)
   (IL:%CML-TYPE-TO-TYPENUMBER-EXPANDER TYPE))

(DEFMACRO MAKE-TEST-DEFUN (TEST-NAME)
   `(DEFUN ,TEST-NAME ()
       ,@(NTHCDR 3 (IL:GETDEF TEST-NAME 'VERIFIED-TESTS))))

(DEFCOMMAND "COPY-TEST" (FROM TO) (IL:COPYDEF FROM TO 'VERIFIED-TESTS))

(DEFCOMMAND "E-TEST" (NAME) (ED NAME '(:DONTWAIT VERIFIED-TESTS)))

(IL:PUTPROPS DO-TEST IL:FILETYPE :COMPILE-FILE)

(IL:PUTPROPS IL:DO-TEST IL:COPYRIGHT ("Venue" 1990))
```

FUNCTION INDEX

MACRO INDEX

VARIABLE INDEX

COMMAND INDEX

DEFINER INDEX

DEFINE-TYPE INDEX

PROPERTY INDEX

OPTIMIZER INDEX