File created: 15-Mar-2024 20:39:04 **{DSK}<home>larry>il>medley>sources>CMLSPECIALFORMS.;4**

edit by: lmm

changes to: (IL:VARS IL:CMLSPECIALFORMSCOMS)

previous date: 15-Mar-2024 10:39:44 {DSK}<home>larry>il>medley>sources>CMLSPECIALFORMS.;2

Read Table: XCL

Package: LISP

Format: XCCS

```
(IL:RPAQQ IL:CMLSPECIALFORMSCOMS
          ((IL:COMS (IL:COMS (IL:FUNCTIONS IDENTITY)
                             (XCL:OPTIMIZERS IDENTITY))
                    (IL:FUNCTIONS UNLESS WHEN))
           (IL:FUNCTIONS FLET LABELS IL:SELECTQ)
           (IL:COMS
                  ;; DO DO* and support.
                  (IL:FUNCTIONS DO DO*)
                  (IL:FUNCTIONS %DO-TRANSLATE))
           (IL:COMS (IL:FUNCTIONS DOLIST DOTIMES)
                  (IL:FUNCTIONS CASE))
           (IL:COMS
                  ;; hacks, These probably shouldn't be here
                  (IL:COMS
                         ;; Hacks for Interlisp NLAMBDAs that should look like functions
                         (IL:PROP IL:MACRO IL:FRPTQ IL:SETN IL:SUB1VAR IL:*))
                  (IL:COMS (IL:FNS IL:BQUOTIFY)
                         (IL:USERMACROS . `IL:UNCOMMA)
                         (IL:VARS IL:*BQUOTE-COMMA* IL:*BQUOTE-COMMA-ATSIGN* IL:*BQUOTE-COMMA-DOT*)
                         (IL:GLOBALVARS IL:*BQUOTE-COMMA* IL:*BQUOTE-COMMA-ATSIGN* IL:*BQUOTE-COMMA-DOT*))
                  (IL:COMS (IL:FNS IL:CLEAR-CLISPARRAY)
                         (IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOCOPY (IL:ADDVARS (IL:MARKASCHANGEDFNS
                                                                               IL:CLEAR-CLISPARRAY))))
                  (IL:P (PROCLAIM '(SPECIAL IL:FILEPKGFLG IL:DFNFLG *READTABLE*))
                        (PROCLAIM (CONS 'SPECIAL IL:SYSSPECVARS))))
           (IL:PROP (IL:FILETYPE IL:MAKEFILE-ENVIRONMENT)
                  IL:CMLSPECIALFORMS)
           (IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE IL:DONTCOPY IL:COMPILERVARS (IL:ADDVARS (IL:NLAMA)
                                                                                                  (IL:NLAML)
                                                                                                  (IL:LAMA)))))

(DEFUN IDENTITY (THING)
   ;; Returns what was passed to it.  Default for :key options.
   THING)

(XCL:DEFOPTIMIZER IDENTITY (X)
                  X)

(DEFMACRO UNLESS (TEST &BODY BODY)
   `(COND
       (,(IL:NEGATE TEST)
        ,@BODY)))

(DEFMACRO WHEN (TEST &BODY BODY)
   `(COND
       (,TEST ,@BODY)))

(DEFMACRO FLET (FUNCTION-BINDINGS &BODY BODY &ENVIRONMENT ENV)

;;; This is only used by the old interpreter and compiler.  The new ones treat FLET specially.

   (LET
    ((FUNCTIONS (MAPCAR #'(LAMBDA (X)
                                  (CONS (GENSYM)
                                        X))
                        FUNCTION-BINDINGS)))
    `(,'LET ,(MAPCAR #'(LAMBDA (X)
                               (XCL:DESTRUCTURING-BIND (FN-NAME FN-ARGLIST &REST FN-BODY)
                                      (CDR X)
                                      (MULTIPLE-VALUE-BIND (BODY DECLS)
                                          (XCL:PARSE-BODY FN-BODY ENV T)
                                        `(,(CAR X)
                                          #'(LAMBDA ,FN-ARGLIST ,@DECLS (BLOCK ,FN-NAME ,@BODY))))))
                     FUNCTIONS)
            ,(XCL:WALK-FORM `(LOCALLY ,@BODY)
```

```
                    :ENVIRONMENT ENV :WALK-FUNCTION
                    #'(LAMBDA (FORM CONTEXT)
                              (IF (OR (ATOM FORM)
                                      (NOT (EQ CONTEXT :EVAL)))
                                  FORM
                                  (COND
                                     ((MEMBER (CAR FORM)
                                              '(IL:FUNCTION FUNCTION)
                                              :TEST
                                              #'EQ)
                                      (DOLIST (Z FUNCTIONS FORM)
                                          (IF (EQ (CADR FORM)
                                                  (CADR Z))
                                              (RETURN (CAR Z)))))
                                     (T (DOLIST (Z FUNCTIONS FORM)
                                            (IF (EQ (CAR FORM)
                                                    (CADR Z))
                                                (RETURN `(FUNCALL ,(CAR Z)
                                                                  ,@(CDR FORM))))))))))
```

(DEFMACRO **LABELS** (FUNCTION-BINDINGS &BODY BODY &ENVIRONMENT ENV)

;;; This is only used by the old interpreter and compiler. The new ones treat LABELS specially.

;;; (Actually, the new compiler still uses this, but it will soon stop doing so.)

```
   (LET
    ((FUNCTIONS (MAPCAR #'(LAMBDA (X)
                                  (CONS (GENSYM)
                                        X))
                        FUNCTION-BINDINGS)))
     `(,'LET
       ,(MAPCAR #'CAR FUNCTIONS)
       ,(XCL:WALK-FORM
         `(PROGN ,@(MAPCAR #'(LAMBDA (X)
                                     (XCL:DESTRUCTURING-BIND
                                      (FN-NAME FN-ARGLIST &REST FN-BODY)
                                      (CDR X)
                                      (MULTIPLE-VALUE-BIND (BODY DECLS)
                                          (XCL:PARSE-BODY FN-BODY ENV T)
                                        `(SETQ ,(CAR X)
                                               #'(LAMBDA ,FN-ARGLIST ,@DECLS (BLOCK ,FN-NAME ,@BODY))))))
                           FUNCTIONS)
                   (LOCALLY ,@BODY))
         :ENVIRONMENT ENV :WALK-FUNCTION
         #'(LAMBDA (FORM CONTEXT)
                   (IF (OR (ATOM FORM)
                           (NOT (EQ CONTEXT :EVAL)))
                       FORM
                       (COND
                          ((MEMBER (CAR FORM)
                                   '(IL:FUNCTION FUNCTION)
                                   :TEST
                                   #'EQ)
                           (DOLIST (Z FUNCTIONS FORM)
                               (IF (EQ (CADR FORM)
                                       (CADR Z))
                                   (RETURN (CAR Z)))))
                          (T (DOLIST (Z FUNCTIONS FORM)
                                 (IF (EQ (CAR FORM)
                                         (CADR Z))
                                     (RETURN `(FUNCALL ,(CAR Z)
                                                       ,@(CDR FORM)))))))))))
```

(DEFMACRO **IL:SELECTQ** (SELECTOR &REST FORMS)

```
   (COND
      ((EQUAL SELECTOR '(IL:SYSTEMTYPE))
       ;; Special case required by the IRM.  (selectq (systemtype) ...) mustn't even look at the untaken arms.
       (LET ((TYPE (EVAL SELECTOR))
             (TAIL FORMS))
            (LOOP (IF (NULL (CDR TAIL))
                      ;; No more possibilities, so use the default.
                      (RETURN (CAR TAIL))
                      ;; Normal clause.  Is this the one we want?
                      (WHEN (OR (EQ TYPE (CAAR TAIL))
                                (AND (CONSP (CAAR TAIL))
                                     (MEMBER TYPE (CAAR TAIL)
                                             :TEST
                                             #'EQ)))
                          (RETURN `(PROGN ,@(CDAR TAIL)))))
                  (SETQ TAIL (CDR TAIL)))))
      (T
```

```
       (LET*
        ((KV (IF (SYMBOLP SELECTOR)
                 SELECTOR
                 (GENSYM)))
         (CLAUSES
          (XCL:WITH-COLLECTION
           (DO ((C FORMS (CDR C)))
               ((NULL C))
             (XCL:COLLECT
              (COND
                 ((NULL (CDR C))
                  `(T ,(CAR C)))
                 ((NOT (CONSP (CAAR C)))
                  `((EQ ,KV ',(CAAR C))
                    ,@(CDAR C)))
                 (T `((OR ,@(MAPCAR #'(LAMBDA (X)
                                        `(EQ ,KV ',X))
                                    (CAAR C)))
                      ,@(CDAR C)))))))))
        (IF (EQ KV SELECTOR)
            `(COND
                ,@CLAUSES)
            `(LET ((,KV ,SELECTOR))
                  (DECLARE (IL:LOCALVARS ,KV))
                  (COND
                     ,@CLAUSES)))))))
```

;; DO DO* and support.

```
(DEFMACRO DO (VARS END-TEST &BODY BODY &ENVIRONMENT ENV)
   (%DO-TRANSLATE VARS END-TEST BODY NIL ENV))


(DEFMACRO DO* (BINDS END-TEST &REST BODY &ENVIRONMENT ENV)
   (%DO-TRANSLATE BINDS END-TEST BODY T ENV))


(DEFUN %DO-TRANSLATE (VARS END-TEST BODY SEQUENTIALP ENV)
   (LET ((VARS-AND-INITIAL-VALUES (MAPCAR #'(LAMBDA (X)
                                              (IF (CONSP X)
                                                  (LIST (CAR X)
                                                        (CADR X))
                                                  (LIST X NIL)))
                                          VARS))
         (SUBSEQUENT-VALUES (MAPCAN #'(LAMBDA (X)
                                        (AND (CONSP X)
                                             (CDDR X)
                                             `((,(CAR X)
                                                ,(CADDR X)))))
                                    VARS))
         (TAG (GENSYM)))
        (IF SUBSEQUENT-VALUES
            (SETQ SUBSEQUENT-VALUES (CONS (IF SEQUENTIALP
                                              'SETQ
                                              'PSETQ)
                                          (APPLY 'APPEND SUBSEQUENT-VALUES))))
        (MULTIPLE-VALUE-BIND (BODY DECLS)
            (XCL:PARSE-BODY BODY ENV)
          `(,(IF SEQUENTIALP
                 'PROG*
                 'PROG)
            ,VARS-AND-INITIAL-VALUES
            ,@DECLS
            ,TAG
            (COND
               (,(CAR END-TEST)
                (RETURN (PROGN ,@(CDR END-TEST)))))
            ,@BODY
            ,SUBSEQUENT-VALUES
            (GO ,TAG)))))


(DEFMACRO DOLIST ((VAR LISTFORM &OPTIONAL RESULTFORM)
                  &BODY BODY &ENVIRONMENT ENV)
   (LET ((TAIL (GENSYM)))
        (MULTIPLE-VALUE-BIND (BODY DECL)
            (XCL:PARSE-BODY BODY ENV)
          `(,'LET ((,TAIL ,LISTFORM)
                   ,VAR)
                  ,@DECL
                  (LOOP (SETQ ,VAR (CAR (OR ,TAIL ,@(IF RESULTFORM
                                                         `((SETQ ,VAR NIL)))
                                            (RETURN ,RESULTFORM))))
                        ,@BODY
                        (SETQ ,TAIL (CDR ,TAIL)))))))
```

```
(DEFMACRO DOTIMES ((VAR COUNTFORM &OPTIONAL RESULTFORM)
                   &BODY BODY &ENVIRONMENT ENV)
   (LET ((MAX (GENSYM)))
        (MULTIPLE-VALUE-BIND (BODY DECLS)
            (XCL:PARSE-BODY BODY ENV)
          `(LET ((,MAX ,COUNTFORM)
                 (,VAR 0))
                ,@DECLS
                (LOOP (IF (>= ,VAR ,MAX)
                          (RETURN ,RESULTFORM))
                      ,@BODY
                      (SETQ ,VAR (1+ ,VAR)))))))


(DEFMACRO CASE (SELECTOR &REST CASES)
   (LET*
    ((KV (IF (SYMBOLP SELECTOR)
             SELECTOR
             (GENSYM)))
     (CLAUSES
      (MAPCAR
       #'(LAMBDA (CASE)
                 (LET ((KEY-LIST (CAR CASE))
                       (CONSEQUENTS (OR (CDR CASE)
                                        (LIST NIL))))
                      (COND
                         ((MEMBER KEY-LIST '(T OTHERWISE)
                                  :TEST
                                  #'EQ)
                          `(T ,@CONSEQUENTS))
                         ((NULL KEY-LIST)
                          (WARN "~S used as a singleton key in ~S. You probably meant to use (~S)." NIL
                                'CASE NIL)
                          '(NIL))
                         ((ATOM KEY-LIST)
                          `((EQL ,KV ',KEY-LIST)
                            ,@CONSEQUENTS))
                         (T `((OR ,@(MAPCAR #'(LAMBDA (X)
                                                      `(EQL ,KV ',X))
                                            KEY-LIST))
                              ,@CONSEQUENTS)))))
       CASES)))
    (IF (EQ KV SELECTOR)
        `(COND
           ,@CLAUSES)
        `(LET ((,KV ,SELECTOR))
              (COND
                 ,@CLAUSES)))))
```

;; hacks, These probably shouldn't be here

;; Hacks for Interlisp NLAMBDAs that should look like functions

```
(IL:PUTPROPS IL:FRPTQ IL:MACRO (= . IL:RPTQ))

(IL:PUTPROPS IL:SETN IL:MACRO (= . IL:SETQ))

(IL:PUTPROPS IL:SUB1VAR IL:MACRO ((IL:X)
                                  (IL:SETQ IL:X (IL:SUB1 IL:X))))

(IL:PUTPROPS IL:* IL:MACRO ((IL:X . IL:Y)
                            'IL:X))

(IL:DEFINEQ

(IL:BQUOTIFY
  (IL:LAMBDA (IL:FORM)                                    (IL:* IL:|bvm:| "10-Jun-86 17:07")

          (IL:* IL:|turn| IL:FORM IL:|into| IL:\a IL:BQUOTE IL:|if| IL:|it| IL:|can.|
          IL:I\f IL:|so,| IL:|return| IL:|it| IL:|as| IL:\a IL:|list,| IL:|otherwise,| IL:|return| NIL)

    (COND
       ((IL:LISTP IL:FORM)
        (LET ((IL:FN (CAR IL:FORM))
              (IL:TAIL (CDR IL:FORM)))
             (AND (IL:LISTP IL:TAIL)
                  (OR (NULL (CDR IL:TAIL))
                      (AND (IL:LISTP (CDR IL:TAIL))
                           (OR (NULL (CDDR IL:TAIL))
                               (IL:SELECTQ IL:FN
                                     ((CONS IL:NCONC1)    (IL:* "These take exactly two args, so if there are more, it's an
                                                          error")
                                           NIL)
                                     T))))
                  (IL:SELECTQ IL:FN
                        ('IL:BQUOTE
```

```
                              (AND (NULL (CDR IL:TAIL))
                                   (LIST (CAR IL:TAIL))))
                         (LIST (LIST (IL:|for| IL:X IL:|in| IL:TAIL IL:|join| (OR (IL:BQUOTIFY IL:X)
                                                                                  (LIST (LIST IL:*BQUOTE-COMMA* IL:X)))))
)
                         ((CONS LIST*)
                              (LIST (IL:APPEND (OR (IL:BQUOTIFY (CAR IL:TAIL))
                                                   (LIST (LIST IL:*BQUOTE-COMMA* (CAR IL:TAIL))))
                                               (OR (CAR (IL:BQUOTIFY (IL:SETQ IL:TAIL (COND
                                                                                          ((AND (EQ IL:FN
                                                                                                    'LIST*)
                                                                                                (CDDR IL:TAIL))
                                                                                           (CONS 'LIST* (CDR IL:TAIL)
                                                                                                 ))
                                                                                          (T (CADR IL:TAIL))))))
                                                   (LIST (LIST IL:*BQUOTE-COMMA-ATSIGN* IL:TAIL))))))
                         ((IL:APPEND NCONC IL:NCONC1)
                              (LET ((IL:DEFAULT (COND
                                                    ((EQ IL:FN 'IL:APPEND)
                                                     IL:*BQUOTE-COMMA-ATSIGN*)
                                                    (T IL:*BQUOTE-COMMA-DOT*)))
                                    (IL:BQCAR (IL:BQUOTIFY (CAR IL:TAIL))))
                                   (LIST (IL:APPEND (COND
                                                        ((AND IL:BQCAR (IL:|for| (IL:TL IL:_ (IL:SETQ IL:BQCAR
                                                                                              (CAR IL:BQCAR)))
                                                                               IL:|by| (CDR IL:TL) IL:|while| IL:TL
                                                                               IL:|never| (IL:NLISTP IL:TL)))
                                                                          (IL:* "Second condition catches (APPEND (CONS A 0) --), where
                                                                          the (CONS A 0) turns into (,A . 0) and then the APPEND would
                                                                          lose it.  It will lose it at runtime, too, of course, but let's not
                                                                          remove mistakes from the source.")
                                                         IL:BQCAR)
                                                        (T (LIST (LIST IL:DEFAULT (CAR IL:TAIL)))))
                                                    (COND
                                                        ((EQ IL:FN 'IL:NCONC1)
                                                                          (IL:* "Second arg is an element, not a segment")
                                                         (OR (IL:BQUOTIFY (IL:SETQ IL:TAIL (CADR IL:TAIL)))
                                                             (LIST (LIST IL:*BQUOTE-COMMA* IL:TAIL))))
                                                        (T (OR (CAR (IL:BQUOTIFY (IL:SETQ IL:TAIL
                                                                                          (COND
                                                                                             ((CDDR IL:TAIL)
                                                                                              (CONS IL:FN (CDR IL:TAIL)))
                                                                                             (T (CADR IL:TAIL))))))
                                                               (LIST (LIST IL:DEFAULT IL:TAIL)))))))))
                              NIL))))
       ((OR (IL:NUMBERP IL:FORM)
            (IL:STRINGP IL:FORM)
            (EQ IL:FORM T)
            (NULL IL:FORM))
        (LIST IL:FORM))
       (T NIL))))

)

(IL:ADDTOVAR IL:USERMACROS
             (IL:UNCOMMA NIL (IL:IF (EQ (IL:\## 1)
                                        'IL:BQUOTE)
                                    NIL
                                    ((IL:IF (EQ (IL:\## IL:!0 1)
                                                'IL:BQUOTE)
                                            (IL:!0))))
                         (IL:I 2 (IL:\\UNCOMMA (IL:\## 2)))))

(IL:ADDTOVAR IL:EDITMACROS
             (IL:BQUOTE NIL IL:UP (IL:ORR ((IL:I 1 (OR (CONS 'IL:BQUOTE (OR (IL:BQUOTIFY (IL:\## 1))
                                                                            (IL:ERROR!)))
                                                       (IL:ERROR!))))
                                          ((IL:E 'IL:BQUOTE?)))
                         1))

(IL:ADDTOVAR IL:EDITCOMSA IL:BQUOTE IL:UNCOMMA)

(IL:RPAQQ IL:*BQUOTE-COMMA* IL:\\\,)

(IL:RPAQQ IL:*BQUOTE-COMMA-ATSIGN* IL:\\\,@)

(IL:RPAQQ IL:*BQUOTE-COMMA-DOT* IL:\\\,.)

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

(IL:GLOBALVARS IL:*BQUOTE-COMMA* IL:*BQUOTE-COMMA-ATSIGN* IL:*BQUOTE-COMMA-DOT*)
)

(IL:DEFINEQ

(IL:CLEAR-CLISPARRAY
  (IL:LAMBDA (IL:NAME TYPE IL:REASON)                    (IL:* IL:|bvm:| "25-Jun-86 12:59")
```

```
    (IL:SELECTQ IL:REASON
          ((T IL:CLISP)                                    (IL:* "New definition or changed only by CLISP translation")
                NIL)
          (CLRHASH IL:CLISPARRAY))))

)

(IL:DECLARE\: IL:DONTEVAL@LOAD IL:DOCOPY

(IL:ADDTOVAR IL:MARKASCHANGEDFNS IL:CLEAR-CLISPARRAY)
)

(PROCLAIM '(SPECIAL IL:FILEPKGFLG IL:DFNFLG *READTABLE*))

(PROCLAIM (CONS 'SPECIAL IL:SYSSPECVARS))

(IL:PUTPROPS IL:CMLSPECIALFORMS IL:FILETYPE COMPILE-FILE)

(IL:PUTPROPS IL:CMLSPECIALFORMS IL:MAKEFILE-ENVIRONMENT (:READTABLE "XCL" :PACKAGE "LISP"))

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

(IL:ADDTOVAR IL:NLAMA )

(IL:ADDTOVAR IL:NLAML )

(IL:ADDTOVAR IL:LAMA )
)
```

FUNCTION INDEX

MACRO INDEX

VARIABLE INDEX

PROPERTY INDEX

OPTIMIZER INDEX