

File created: 28-Oct-91 11:45:00 {PELE:MV:ENVOS}<LISPCORE>SOURCES>CMLMACROS.;4

changes to: (FNS CL:MACRO-FUNCTION)

previous date: 18-Sep-90 17:33:42 {PELE:MV:ENVOS}<LISPCORE>SOURCES>CMLMACROS.;3

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;
;; Copyright (c) 1986, 1987, 1990, 1991 by Venue & Xerox Corporation. All rights reserved.

(RPAQQ CMLMACROSCOMS

```
[ (FNS CLISPEXPANSION GLOBAL-MACRO-FUNCTION LOCAL-MACRO-FUNCTION LOCAL-SYMBOL-FUNCTION
  \INTERLISP-NLAMBDA-MACRO CL:MACRO-FUNCTION CL:MACROEXPAND CL:MACROEXPAND-1 SETF-MACRO-FUNCTION)
  (APPENDVARS (COMPIILERMACROPROPS DMACRO BYTEMACRO MACRO))
  (ADDVARS (GLOBALVARS COMPIILERMACROPROPS))
  (PROP MACRO *)
  (FUNCTIONS CL:MACROLET)
  (SETFS CL:MACRO-FUNCTION)
  (PROP FILETYPE CMLMACROS)
  (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPIILERVARS (ADDVARS (NLAMA)
                                                                    (NLAML)
                                                                    (LAMA]))
```

(DEFINEQ

(CLISPEXPANSION

[LAMBDA (X ENV)

; Edited 4-Dec-86 01:19 by Imm

;; the macro function for all CLISP words. Expand X as a clisp macro.

```
(CL:VALUES (do (LET ((NOSPPELLFLG T)
                    (LISPXHIST NIL)
                    (VARS NIL)
                    (COP (COPY X)))
              (DECLARE (CL:SPECIAL NOSPPELLFLG VARS LISPXHIST))
              ; make a copy so dwim doesn't muck with it!
              [COND
                ((GETPROP (CAR X)
                          'CLISPWORD)
                 (DWIMIFY0? COP COP COP NIL NIL NIL 'VARSBOUND)
                 (COND
                  ((NOT (CL:EQUAL COP X)) ; made a change
                   (RETURN COP))
                  ((SETQ COP (GETHASH COP CLISPARRAY))
                   (RETURN COP))
                  (CL:CERROR "Try expanding again." "Can't CLISP expand expression ~S." X)))
              T]))
```

(GLOBAL-MACRO-FUNCTION

[LAMBDA (X ENV)

; Edited 22-Apr-87 19:07 by Pavel

```
(LET (MD)
  (COND
   [(AND (TYPEP ENV 'COMPIILER:ENV)
         (CL:MULTIPLE-VALUE-BIND (KIND EXPN-FN)
                                  (COMPIILER:ENV-FBOUNDP ENV X)
                                  (AND (EQ KIND :MACRO)
                                       EXPN-FN))]
        ((GET X 'MACRO-FN))
        ((CL:SPECIAL-FORM-P X)
         NIL)
        [[AND [NOT (FMEMB (ARGTYPE X)
                          '(0 2))
              (FIND PROP IN COMPIILERMACROPROPS SUCHTHAT (AND (SETQ MD (GETPROP X PROP))
                                                                (NOT (OR (LITATOM MD)
                                                                (FMEMB (CAR MD)
                                                                '(APPLY APPLY*])
                                                                ' (LAMBDA (FORM ENV)
                                                                (MACROEXPANSION FORM ' ,MD]
                                                                ((AND (NOT (GETD X))
                                                                (GETPROP X 'CLISPWORD))
                                                                (FUNCTION CLISPEXPANSION))
                                                                ((FMEMB (ARGTYPE X)
                                                                '(1 3))
                                                                (FUNCTION \INTERLISP-NLAMBDA-MACRO))
```

(LOCAL-MACRO-FUNCTION

[LAMBDA (X ENV)

; Edited 13-Apr-87 11:16 by Pavel

```
(AND ENV (CL:TYPECASE ENV
              (ENVIRONMENT ; Interpreter's environments
               (LET ((FN-DEFN (CL:GETF (ENVIRONMENT-FUNCTIONS ENV)
```

```

      X)))
      (AND FN-DEFN (EQ (CAR FN-DEFN)
                      :MACRO)
            (CDR FN-DEFN)))
      (COMPILER:ENV ; Compiler's environments.
      (CL:MULTIPLE-VALUE-BIND (KIND EXPN-FN)
      (COMPILER:ENV-FBOUNDP ENV X :LEXICAL-ONLY T)
      (AND (EQ KIND :MACRO)
            EXPN-FN))))])

```

(LOCAL-SYMBOL-FUNCTION

```

[LAMBDA (X ENV) ; Edited 31-Jul-87 18:06 by amd
  (AND ENV (CL:TYPECASE ENV
            (ENVIRONMENT ; Interpreter's environments
              (LET ((FN-DEFN (CL:GETF (ENVIRONMENT-FUNCTIONS ENV)
                                     X)))
                  (AND FN-DEFN (EQ (CAR FN-DEFN)
                                  :FUNCTION)
                        (CDR FN-DEFN))))
                (COMPILER:ENV ; Compiler's environments.
                (CL:MULTIPLE-VALUE-BIND (KIND FN)
                (COMPILER:ENV-FBOUNDP ENV X :LEXICAL-ONLY T)
                (AND (EQ KIND :FUNCTION)
                    FN))))))])

```

(INTERLISP-NLAMBDA-MACRO

```

[LAMBDA (X ENV) (* Imm " 7-May-86 17:24")
  `(CL:FUNCALL (FUNCTION , (CAR X))
              ,@(SELECTQ (ARGTYPE (CAR X))
                        (1 (MAPCAR (CDR X)
                                  (FUNCTION KWOTE))))
                (3 (LIST (KWOTE (CDR X))))
                  (SHOULDNT]))

```

(CL:MACRO-FUNCTION

```

[CL:LAMBDA (CL::X CL::ENV) ; Edited 28-Oct-91 11:44 by jds
  (AND (CL:SYMBOLP CL::X)
        (NOT (LOCAL-SYMBOL-FUNCTION CL::X CL::ENV))
        (OR (LOCAL-MACRO-FUNCTION CL::X CL::ENV)
            (GLOBAL-MACRO-FUNCTION CL::X CL::ENV]))

```

(CL:MACROEXPAND

```

[CL:LAMBDA (CL::FORM &OPTIONAL CL::ENV) ; Edited 13-Feb-87 23:47 by Pavel

```

;;; If FORM is a macro call, then the form is expanded until the result is not a macro. Returns as multiple values, the form after any expansion has been done and T if expansion was done, or NIL otherwise. Env is the lexical environment to expand in, which defaults to the null environment.

```

(PROG (CL::FLAG)
  (CL:MULTIPLE-VALUE-SETQ (CL::FORM CL::FLAG)
                        (CL:MACROEXPAND-1 CL::FORM CL::ENV))
  (CL:UNLESS CL::FLAG
    (RETURN (CL:VALUES CL::FORM NIL)))
  CL:LOOP
  (CL:MULTIPLE-VALUE-SETQ (CL::FORM CL::FLAG)
                        (CL:MACROEXPAND-1 CL::FORM CL::ENV))
  (CL:IF CL::FLAG
    (GO CL:LOOP)
    (RETURN (CL:VALUES CL::FORM T))))

```

(CL:MACROEXPAND-1

```

[CL:LAMBDA (CL::FORM &OPTIONAL CL::ENV) ; Edited 13-Feb-87 23:49 by Pavel

```

;;; If form is a macro, expands it once. Returns two values, the expanded form and a T-or-NIL flag indicating whether the form was, in fact, a macro. Env is the lexical environment to expand in, which defaults to the null environment.

```

(COND
  [(AND (CL:CONSP CL::FORM)
        (CL:SYMBOLP (CAR CL::FORM)))
    (LET ((CL::DEF (CL:MACRO-FUNCTION (CAR CL::FORM)
                                     CL::ENV))
          (COND
            (CL::DEF (CL:IF [NOT (EQ CL::FORM (CL:SETQ CL::FORM (CL:FUNCALL *MACROEXPAND-HOOK* CL::DEF
                                                                              CL::FORM CL::ENV)]
                                (CL:VALUES CL::FORM T)
                                (CL:VALUES CL::FORM NIL))))
              (T (CL:VALUES CL::FORM NIL])
              (T (CL:VALUES CL::FORM NIL]))

```

(SETF-MACRO-FUNCTION

```

[LAMBDA (X BODY) ; Edited 13-Feb-87 13:26 by Pavel

```

;; the SETF function for MACRO-FUNCTION

;; NOTE: If you change this, be sure to change the undoable version on CMLUNDO!

```

(PROG1 (CL:SETF (GET X 'MACRO-FN)
              BODY)
      (AND (GETD X)
           (SELECTQ (ARGTYPE X)
                    ((1 3)
                     )
                    (PUTD X NIL))))))
)
; Leave Interlisp nlambda definition alone

(APPENDTOVAR COMPILERMACROPROPS DMACRO BYTEMACRO MACRO)

(ADDTOVAR GLOBALVARS COMPILERMACROPROPS)

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

(DEFMACRO CL:MACROLET (CL::MACRODEFS &BODY CL::BODY &ENVIRONMENT CL::ENV)
  (DECLARE (SPECVARS *BYTECOMPILER-IS-EXPANDING*))
  ;; This macro for the old interpreter and compiler only. The new interpreter has a special-form definition. When the new compiler is expanding, we
  ;; simply return a disguised version of the form.
  (IF (AND *BYTECOMPILER-IS-EXPANDING* *BYTECOMPILER-OPTIMIZE-MACROLET*)
      THEN (LET ((CL::NEW-ENV (COMPILER::MAKE-CHILD-ENV CL::ENV)))
             (DECLARE (CL:SPECIAL *BC-MACRO-ENVIRONMENT*))
             [FOR CL::FN IN CL::MACRODEFS DO (COMPILER::ENV-BIND-FUNCTION CL::NEW-ENV (CAR CL::FN)
                                   :MACRO
                                   (COMPILER::CRACK-DEFMACRO (CONS 'DEFMACRO CL::FN)
                                                             (CL:SETQ *BC-MACRO-ENVIRONMENT* CL::NEW-ENV)
                                                             (CONS 'CL:LOCALLY CL::BODY))
                                   )
              ]
      ELSEIF (TYPEP CL::ENV 'COMPILER:ENV)
      THEN `(SI::%%MACROLET ,CL::MACRODEFS ,@CL::BODY)
      ELSE (LET* ((CL::NEW-ENV (\MAKE-CHILD-ENVIRONMENT CL::ENV))
                 (CL::FUNCTIONS (ENVIRONMENT-FUNCTIONS CL::NEW-ENV)))
             (FOR CL::FN IN CL::MACRODEFS
              DO (CL:SETQ CL::FUNCTIONS (LIST* (CAR CL::FN)
                                               [CONS :MACRO
                                                  `(CL:LAMBDA (SI::$$MACRO-FORM SI::$$MACRO-ENVIRONMENT)
                                                    (CL:BLOCK , (CAR CL::FN)
                                                      , (PARSE-DEFMACRO (CADR CL::FN)
                                                                    'SI::$$MACRO-FORM
                                                                    (CDDR CL::FN)
                                                                    (CAR CL::FN)
                                                                    NIL :ENVIRONMENT
                                                                    'SI::$$MACRO-ENVIRONMENT))
                                                    ]
                                               )
              )
             (CL:SETF (ENVIRONMENT-FUNCTIONS CL::NEW-ENV)
                     CL::FUNCTIONS)
             (WALK-FORM (CONS 'CL:LOCALLY CL::BODY)
                       :ENVIRONMENT CL::NEW-ENV)))
  )

(CL:DEFSETF CL:MACRO-FUNCTION SETF-MACRO-FUNCTION)

(PUTPROPS CMLMACROS FILETYPE CL:COMPILE-FILE)

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS)

(ADDTOVAR NLAMA )
(ADDTOVAR NLAML )
(ADDTOVAR LAMA )
)

(RPAQQ CMLMACROSCOMS
  [(FNS CLISPEXPANSION GLOBAL-MACRO-FUNCTION LOCAL-MACRO-FUNCTION LOCAL-SYMBOL-FUNCTION
        \INTERLISP-NLAMBDA-MACRO CL:MACRO-FUNCTION CL:MACROEXPAND CL:MACROEXPAND-1 SETF-MACRO-FUNCTION)
   (APPENDVARS (COMPILERMACROPROPS DMACRO BYTEMACRO MACRO))
   (ADDVARS (GLOBALVARS COMPILERMACROPROPS))
   (PROP MACRO *)
   (FUNCTIONS CL:MACROLET)
   (SETFS CL:MACRO-FUNCTION)
   (PROP FILETYPE CMLMACROS)
   (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                    (NLAML)
                                                                    (LAMA CL:MACRO-FUNCTION))
  )

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS)

(ADDTOVAR NLAMA )

```

{MEDLEY}<sources>CMLMACROS.;1

(ADDTOVAR **NLAML**)

(ADDTOVAR **LAMA** CL:MACRO-FUNCTION)
)

(PUTPROPS **CMLMACROS COPYRIGHT** ("Venue & Xerox Corporation" 1986 1987 1990 1991))

FUNCTION INDEX

CLISPEXPANSION	1	LOCAL-SYMBOL-FUNCTION	2	CL:MACROEXPAND-1	2
GLOBAL-MACRO-FUNCTION	1	CL:MACRO-FUNCTION	2	SETF-MACRO-FUNCTION	2
LOCAL-MACRO-FUNCTION	1	CL:MACROEXPAND	2	\INTERLISP-NLAMBDA-MACRO	2

MACRO INDEX

*	3	CL:MACROLET	3
---------	---	-------------------	---

PROPERTY INDEX

CMLMACROS	3
-----------------	---

SETF INDEX

CL:MACRO-FUNCTION	3
-------------------------	---
