```
    File created:   16-May-90 21:15:43  {DSK}<usr>local>lde>lispcore>sources>PROFILE.;2

      changes to:   (IL:VARS IL:PROFILECOMS)

   previous date:   27-Feb-87 14:34:18  {DSK}<usr>local>lde>lispcore>sources>PROFILE.;1

      Read Table:   XCL

         Package:   XEROX-COMMON-LISP

           Format:    XCCS
```

```
(IL:RPAQQ IL:PROFILECOMS
           (;; The profile type
            (IL:DEFINE-TYPES PROFILES)
            (IL:FUNCTIONS DEFPROFILE)
            (IL:TYPES PROFILE)
            (IL:STRUCTURES PROFILE-CLAUSE VARIABLE-DEFINITION)
            (IL:VARIABLES *PROFILE* *PROFILE-NAME* *PROFILE-VARIABLES* *PROFILES*)
            (IL:FUNCTIONS FIND-VARIABLE-DEFINITION IN-PROFILE INSTALL-PROFILE MAKE-VARIABLE-DEFINITION PROFILIZE
                   PROFILE-ENTRY-VALUE PROFILE-ENTRY-VALUE-NAME PROFILE-NAME PROFILE-P PROFILE-VALUE-TYPE-CHECK
                   SETF-PROFILE-ENTRY-VALUE SETF-PROFILE-ENTRY-VALUE-NAME SETF-PROFILE-NAME MAKE-PROFILE
                   COPY-PROFILE RESTORE-PROFILE SAVE-PROFILE WITH-PROFILE FIND-PROFILE SETF-FIND-PROFILE
                   LIST-ALL-PROFILES PROFILE-VALUES PROFILE-VARIABLES)
            (IL:SETFS FIND-PROFILE PROFILE-ENTRY-VALUE PROFILE-ENTRY-VALUE-NAME PROFILE-NAME)
            (PROFILES "READ-PRINT" "LISP" "INTERLISP" "OLD-INTERLISP-T" "XEROX-COMMON-LISP")
            (IL:DECLARE\: IL:DONTCOPY IL:DONTEVAL@LOAD IL:DOEVAL@COMPILE (IL:LOCALVARS . T))
            (IL:PROP (IL:MAKEFILE-ENVIRONMENT IL:FILETYPE)
                   IL:PROFILE)))

;; The profile type


(DEF-DEFINE-TYPE PROFILES "interaction profiles")

(DEFDEFINER (DEFPROFILE (:NAME (LAMBDA (WHOLE)
                                          (LET ((NAME-CLAUSE (SECOND WHOLE)))
                                               (IF (CONSP NAME-CLAUSE)
                                                   (STRING (CAR NAME-CLAUSE))
                                                   (STRING NAME-CLAUSE)))))))
    PROFILES (NAME-CLAUSE &REST VARIABLE-CLAUSES)
   "Creates a new named profile. name . clauses or (name (:nicknames n1 n2...)) clauses"
   (LET ((NAME (IF (CONSP NAME-CLAUSE)
                   (STRING (CAR NAME-CLAUSE))
                   (STRING NAME-CLAUSE)))
         (NICKNAMES (AND (CONSP NAME-CLAUSE)
                         (MAPCAR #'STRING (CDADR NAME-CLAUSE)))))
        `(LET ((PROFILE (MAKE-PROFILE ,NAME ,@(MAPCAR #'(LAMBDA (CLAUSE)
                                                                `',CLAUSE)
                                                      VARIABLE-CLAUSES))))
              (INSTALL-PROFILE PROFILE ,NAME ',NICKNAMES))))


(DEFTYPE PROFILE ()
   '(SATISFIES PROFILE-P))


(DEFSTRUCT (PROFILE-CLAUSE (:TYPE LIST)
                           (:CONSTRUCTOR NIL))
   VARIABLE
   NAME
   TYPE
   COERCION-FUNCTION
   NAME-FUNCTION)


(DEFSTRUCT (VARIABLE-DEFINITION (:TYPE LIST)
                                (:CONSTRUCTOR NIL))
   VARIABLE
   TYPE
   COERCION-FUNCTION
   NAME-FUNCTION)


(DEFPARAMETER *PROFILE* "XCL"
   "The default or current profile.")


(DEFPARAMETER *PROFILE-NAME* NIL)


(DEFPARAMETER *PROFILE-VARIABLES*
   '((*PROFILE-NAME* T IDENTITY IDENTITY)
     (*EVAL-FUNCTION* (MEMBER IL:EVAL EVAL)
             IDENTITY IDENTITY)
      (*EXEC-PROMPT* STRING STRING IDENTITY)
      (*DEBUGGER-PROMPT* STRING STRING IDENTITY)
      (*READTABLE* READTABLE IL:FIND-READTABLE IL:READTABLE-NAME)
      (*READ-BASE* (INTEGER 2 36)
             IDENTITY IDENTITY)
      (*READ-SUPPRESS* (MEMBER NIL T)
             IDENTITY IDENTITY)
      (*PACKAGE* PACKAGE FIND-PACKAGE PACKAGE-NAME)
      (*READ-DEFAULT-FLOAT-FORMAT* (MEMBER SINGLE-FLOAT DOUBLE-FLOAT LONG-FLOAT SHORT-FLOAT)
             IDENTITY IDENTITY)
      (*PRINT-ESCAPE* (MEMBER NIL T)
             IDENTITY IDENTITY)
      (*PRINT-PRETTY* (MEMBER NIL T)
             IDENTITY IDENTITY)
      (*PRINT-CIRCLE* (MEMBER NIL T)
             IDENTITY IDENTITY)
      (*PRINT-BASE* (INTEGER 2 36)
             IDENTITY IDENTITY)
      (*PRINT-RADIX* (MEMBER NIL T)
             IDENTITY IDENTITY)
      (*PRINT-CASE* (MEMBER :DOWNCASE :UPCASE :CAPITALIZE)
             IDENTITY IDENTITY)
      (*PRINT-GENSYM* (MEMBER NIL T)
             IDENTITY IDENTITY)
      (*PRINT-LEVEL* (OR NULL FIXNUM)
             IDENTITY IDENTITY)
      (*PRINT-LENGTH* (OR NULL FIXNUM)
             IDENTITY IDENTITY)
      (*PRINT-ARRAY* (MEMBER NIL T)
             IDENTITY IDENTITY)
      (*PRINT-STRUCTURE* (MEMBER NIL T)
             IDENTITY IDENTITY)))


(DEFGLOBALVAR *PROFILES* (MAKE-HASH-TABLE :TEST 'EQUAL)
                           "Where profiles live.")


(DEFUN FIND-VARIABLE-DEFINITION (VARIABLE)
   (DOLIST (ENTRY *PROFILE-VARIABLES* NIL)
       (IF (EQ VARIABLE (VARIABLE-DEFINITION-VARIABLE ENTRY))
           (RETURN ENTRY))))


(DEFUN IN-PROFILE (PROFILE)
   "Makes profile the current profile and resets *profile*"
   (SETQ *PROFILE* (PROFILIZE PROFILE))
   (RESTORE-PROFILE *PROFILE*))


(DEFUN INSTALL-PROFILE (PROFILE PROFILE-NAME PROFILE-NICKNAMES)
   (DOLIST (NAME (CONS PROFILE-NAME PROFILE-NICKNAMES))           ; Make the name and all nicknames point at the new profile.
       (IF (AND (FIND-PROFILE NAME)
                (FBOUNDP 'WARN))
           (WARN "Resetting profile ~s." NAME))
       (SETF (FIND-PROFILE NAME)
             PROFILE)))


(DEFUN MAKE-VARIABLE-DEFINITION (CLAUSE &AUX (DEFINITION NIL))
   "Add a new profile variable entry based on clauses.  clauses is bounded by a keyword or nil."
   (DOLIST (DEFAULT '(NIL                                        ; variable
                      IGNORE                                     ; value
                      T                                          ; type
                      IDENTITY                                   ; coercion-function
                      IDENTITY                                   ; name-function

                      )                                          ; Defaults for a definition's slots.

                  )                                              ; Maps on defaults to always fill all the slots.
       (IF (EQ DEFAULT 'IGNORE)
           (POP CLAUSE)                                          ; Ignore the value slot
           (PUSH                                                 ; Push onto the new entry
                 (IF (NULL CLAUSE)                               ; If we're at the end of the clause:
                     DEFAULT                                     ; ...use the default.
                     (POP CLAUSE)                                ; ...otherwise use the next element of the clause.

                     )
                 DEFINITION)))
   (SETQ DEFINITION (NREVERSE DEFINITION))
   (PUSH DEFINITION                                              ; Flip the push built list.
         *PROFILE-VARIABLES*)                                    ; Put the new definition onto the global list of definitions.
   DEFINITION)


(DEFUN PROFILIZE (NAME-OR-PROFILE)
   (ETYPECASE NAME-OR-PROFILE
       ((OR STRING SYMBOL) (OR (FIND-PROFILE NAME-OR-PROFILE)
                               (ERROR "Not the name of an existing profile ~s" NAME-OR-PROFILE)))
       (PROFILE NAME-OR-PROFILE)))


(DEFUN PROFILE-ENTRY-VALUE (VARIABLE &OPTIONAL (PROFILE *PROFILE*))
   "Returns the value of the variable in the current profile or its binding."
   (GETF (PROFILIZE PROFILE)
         VARIABLE
         (EVAL VARIABLE)))


(DEFUN PROFILE-ENTRY-VALUE-NAME (VARIABLE &OPTIONAL (PROFILE *PROFILE*))
   "Get the name of the value in a variable or the name of the current binding."
   (FUNCALL (VARIABLE-DEFINITION-NAME-FUNCTION (FIND-VARIABLE-DEFINITION VARIABLE))
          (GETF (PROFILIZE PROFILE)
                VARIABLE
                (EVAL VARIABLE))))


(DEFUN PROFILE-NAME (&OPTIONAL (PROFILE *PROFILE*))
   "Returns the name of the profile as a string."
   (PROFILE-ENTRY-VALUE '*PROFILE-NAME* (PROFILIZE PROFILE)))


(DEFUN PROFILE-P (OBJECT)
   "Returns true if the object seems to be a profile.  Is true only of profiles, never their names."
   (AND (CONSP OBJECT)
        (SYMBOLP (FIRST OBJECT))
        (EVENP (LENGTH OBJECT))
        T))


(DEFUN PROFILE-VALUE-TYPE-CHECK (DEFINITION VALUE)
   "Returns correct or corrected value."
   (LET ((COERCION-FUNCTION (VARIABLE-DEFINITION-COERCION-FUNCTION DEFINITION))
         (TYPE (VARIABLE-DEFINITION-TYPE DEFINITION)))
        (LOOP (IF (TYPEP VALUE TYPE)                                   ; Is it of the right type?
                  (RETURN VALUE)                                       ; ...just return it

                  )
              (COND
                 (COERCION-FUNCTION (SETQ VALUE (FUNCALL COERCION-FUNCTION VALUE))
                                                                       ; Perhaps it was a name, coerce it.
                        (IF (TYPEP VALUE TYPE)                         ; Is it NOW of the right type?
                            (RETURN VALUE)                             ; ...just return it

                            )))
              ;; Otherwise we were given something that can't either use or coerce; complain, fix and retry
              (CERROR "Give new value" "Profile slot ~s's value ~s not a(n) ~s" (VARIABLE-DEFINITION-VARIABLE
                                                                                 DEFINITION)
                     VALUE TYPE)
              (FORMAT *QUERY-IO* "Give new value expression (will be evaluated)~%")
              (SETQ VALUE (EVAL (READ))))))


(DEFUN SETF-PROFILE-ENTRY-VALUE (VARIABLE PROFILE VALUE)
   (SETQ PROFILE (PROFILIZE PROFILE))
   (LET ((TYPE (VARIABLE-DEFINITION-TYPE (FIND-VARIABLE-DEFINITION VARIABLE))))
        (ASSERT (TYPEP VALUE TYPE)
               (VALUE)
               "Profile slot ~s's value ~s not a(n) ~s" VARIABLE VALUE TYPE)
        (SETF (GETF PROFILE VARIABLE)
              VALUE)))


(DEFUN SETF-PROFILE-ENTRY-VALUE-NAME (VARIABLE PROFILE NAME)
   (SETQ PROFILE (PROFILIZE PROFILE))
   (SETF (PROFILE-ENTRY-VALUE VARIABLE PROFILE)
         (FUNCALL (VARIABLE-DEFINITION-COERCION-FUNCTION (FIND-VARIABLE-DEFINITION VARIABLE))
                NAME)))


(DEFUN SETF-PROFILE-NAME (PROFILE NAME)
   (SETF (PROFILE-ENTRY-VALUE '*PROFILE-NAME* PROFILE)
         (STRING NAME)))


(DEFUN MAKE-PROFILE (PROFILE-NAME &REST CLAUSES)
   "Creates a profile with slots described by the clauses.  Clauses is an alist of variables and values, similar
   to defstruct's."
   (LET ((PROFILE NIL))
        (DOLIST (CLAUSE CLAUSES)
            (LET* ((VARIABLE (PROFILE-CLAUSE-VARIABLE CLAUSE))
                   (NAME (PROFILE-CLAUSE-NAME CLAUSE))                ; Name of the value to be used (or the value itself).
```

```
                    (DEFINITION (OR (FIND-VARIABLE-DEFINITION VARIABLE)
                                    (MAKE-VARIABLE-DEFINITION CLAUSE))))
                   ;; These are pushed on in reverse order so the final prop list format of the profile will be correct.
                   (PUSH (PROFILE-VALUE-TYPE-CHECK DEFINITION (EVAL NAME))
                         PROFILE)
                   (PUSH VARIABLE PROFILE)))
         (CONS '*PROFILE-NAME* (CONS PROFILE-NAME PROFILE))))


(DEFUN COPY-PROFILE (&OPTIONAL (PROFILE *PROFILE*))
   "Copies the given profile."
   (COPY-SEQ (PROFILIZE PROFILE)))


(DEFUN RESTORE-PROFILE (&OPTIONAL (PROFILE *PROFILE*))
   "Set profile variables from given profile."
   (SETQ PROFILE (PROFILIZE PROFILE))
   (MAPC #'SET (PROFILE-VARIABLES PROFILE)
         (PROFILE-VALUES PROFILE))
   PROFILE)


(DEFUN SAVE-PROFILE (&OPTIONAL (PROFILE *PROFILE*))
   "Save current values of bindings into profile."
   (IL:FOR X IL:ON (PROFILIZE PROFILE) IL:BY CDDR IL:DO (SETF (CADR X)
                                                               (EVAL (CAR X))))
   PROFILE)


(DEFMACRO WITH-PROFILE (PROFILE-FORM &BODY FORMS)
   "Bind all the special IO variables to the values in the profile and execute the body forms."
   `(LET ((*PROFILE* ,PROFILE-FORM))
         (SETQ *PROFILE* (PROFILIZE *PROFILE*))
         (PROGV (PROFILE-VARIABLES *PROFILE*)
                (PROFILE-VALUES *PROFILE*)
                ,@FORMS)))


(DEFUN FIND-PROFILE (NAME)
   (GETHASH (STRING NAME)
          *PROFILES*))


(DEFUN SETF-FIND-PROFILE (NAME PROFILE)
   (SETQ NAME (STRING NAME))
   (SETQ PROFILE (PROFILIZE PROFILE))
   (SETF (GETHASH NAME *PROFILES*)
         PROFILE)
   (SETF (PROFILE-NAME PROFILE)
         NAME)
   NAME)


(DEFUN LIST-ALL-PROFILES ()
   (LET ((PROFILES NIL))
        (MAPHASH #'(LAMBDA (NAME VALUE)
                          (PUSHNEW VALUE PROFILES :TEST #'EQ)      ; Avoid repeats due to nicknames

                          )
               *PROFILES*)
        (MAPCAR #'(LAMBDA (PROFILE)
                         (PROFILE-NAME PROFILE)                    ; Convert to name strings

                         )
               PROFILES)))


(DEFUN PROFILE-VALUES (PROFILE)
   (IL:FOR X IL:IN (CDR (PROFILIZE PROFILE)) IL:BY CDDR IL:COLLECT X))


(DEFUN PROFILE-VARIABLES (&OPTIONAL (PROFILE *PROFILE*))
   (IL:FOR X IL:IN (PROFILIZE PROFILE) IL:BY CDDR IL:COLLECT X))


(DEFSETF FIND-PROFILE SETF-FIND-PROFILE)


(DEFSETF PROFILE-ENTRY-VALUE SETF-PROFILE-ENTRY-VALUE)


(DEFSETF PROFILE-ENTRY-VALUE-NAME SETF-PROFILE-ENTRY-VALUE-NAME)


(DEFSETF PROFILE-NAME SETF-PROFILE-NAME)
```

```
(DEFPROFILE "READ-PRINT" (*READTABLE* "LISP")
                         (*READ-BASE* 10)
                         (*READ-SUPPRESS* NIL)
                         (*PACKAGE* "USER")
                         (*READ-DEFAULT-FLOAT-FORMAT* 'SINGLE-FLOAT)
                         (*PRINT-ESCAPE* T)
                         (*PRINT-PRETTY* NIL)
                         (*PRINT-CIRCLE* NIL)
                         (*PRINT-BASE* 10)
                         (*PRINT-RADIX* NIL)
                         (*PRINT-CASE* :UPCASE)
                         (*PRINT-GENSYM* T)
                         (*PRINT-LEVEL* NIL)
                         (*PRINT-LENGTH* NIL)
                         (*PRINT-ARRAY* NIL)
                         (*PRINT-STRUCTURE* NIL))

(DEFPROFILE ("LISP" (:NICKNAMES "CL")) (*READTABLE* "LISP")
                                       (*PACKAGE* "USER")
                                       (*EVAL-FUNCTION* 'EVAL)
                                       (*EXEC-PROMPT* "> ")
                                       (*DEBUGGER-PROMPT* ": "))

(DEFPROFILE ("INTERLISP" (:NICKNAMES "IL")) (*READTABLE* "INTERLISP")
                                            (*PACKAGE* "INTERLISP")
                                            (*EVAL-FUNCTION* 'IL:EVAL)
                                            (*EXEC-PROMPT* "_ ")
                                            (*DEBUGGER-PROMPT* "_: "))

(DEFPROFILE "OLD-INTERLISP-T" (*READTABLE* "OLD-INTERLISP-T")
                              (*PACKAGE* "INTERLISP")
                              (*EVAL-FUNCTION* 'IL:EVAL)
                              (*EXEC-PROMPT* "_ ")
                              (*DEBUGGER-PROMPT* "_: "))

(DEFPROFILE ("XEROX-COMMON-LISP" (:NICKNAMES "XCL")) (*READTABLE* "XCL")
                                                     (*PACKAGE* "XCL-USER")
                                                     (*EVAL-FUNCTION* 'EVAL)
                                                     (*EXEC-PROMPT* "> ")
                                                     (*DEBUGGER-PROMPT* ": "))

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

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

(IL:LOCALVARS . T)
)
)

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

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

(IL:PUTPROPS IL:PROFILE IL:COPYRIGHT ("Venue & Xerox Corporation" 1986 1987 1990))
```

{MEDLEY}<sources>PROFILE.;1 28-Jun-2024 18:34:03
-- Listed on 30-Jun-2024 13:16:12 --

FUNCTION INDEX

PROFILE INDEX

SETF INDEX

VARIABLE INDEX

STRUCTURE INDEX

PROPERTY INDEX

TYPE INDEX

MACRO INDEX

DEFINER INDEX

DEFINE-TYPE INDEX