File created: 16-May-90 15:39:16 **{DSK}<usr>local>lde>lispcore>sources>DESCRIBE.;2**

changes to: (IL:VARS IL:DESCRIBECOMS)

previous date: 16-May-88 17:04:26 {DSK}<usr>local>lde>lispcore>sources>DESCRIBE.;1

Read Table: XCL

Package: SYSTEM

Format: XCCS


```
(IL:RPAQQ IL:DESCRIBECOMS
          (;; Common LISP describe facility
           (IL:FUNCTIONS DESCRIBE DESCRIBE-INTERNAL DESCRIBE-NEW-LINE DESCRIBE-USING-DESCRIBERS
                  GET-SUPER-DESCRIBERS GET-INSPECT-MACRO INSPECT-MACRO-USABLE-BY-DESCRIBE?
                  DESCRIBE-USING-INSPECT-MACRO DESCRIBE-USING-RECORD-DECL)
           (IL:FUNCTIONS A-OR-AN VOWEL-P)
           (IL:DEFINE-TYPES CL::DESCRIBERS)
           (IL:FUNCTIONS CL::DEFDESCRIBER GET-DESCRIBERS)
           (IL:PROP IL:PROPTYPE DESCRIBERS)
           (CL::DESCRIBERS SYMBOL CL::STRUCTURE-OBJECT CHARACTER FIXNUM SINGLE-FLOAT HASH-TABLE)
           (IL:VARIABLES CL::*DESCRIBE-DEPTH* CL::*DESCRIBE-INDENT* CL::*DESCRIBE-PRINT-LENGTH*
                  CL::*DESCRIBE-PRINT-LEVEL*)
           (IL:PROPS (IL:DESCRIBE IL:MAKEFILE-ENVIRONMENT IL:FILETYPE))))
```

;; Common LISP describe facility

```
(DEFUN DESCRIBE (CL::OBJECT)
   "Describe OBJECT, printing to *STANDARD-OUTPUT*."
   (LET ((*PRINT-LENGTH* CL::*DESCRIBE-PRINT-LENGTH*)
         (*PRINT-LEVEL* CL::*DESCRIBE-PRINT-LEVEL*))
        (DESCRIBE-INTERNAL CL::OBJECT 0)
        (VALUES)))

(DEFUN DESCRIBE-INTERNAL (OBJECT DEPTH)
```

;;; Recursive entry point for descriptions.

```
   (IF (< DEPTH CL::*DESCRIBE-DEPTH*)
       (LET* ((TYPE (TYPE-OF OBJECT))
              (TYPE-NAME (IF (CONSP TYPE)
                             (CAR TYPE)
                             TYPE))
              DESCRIBERS INSPECT-MACRO SUPER-DESCRIBERS IL:DECL)
             (DESCRIBE-NEW-LINE DEPTH)
             (FORMAT T "~A ~A, " (A-OR-AN TYPE-NAME)
                     TYPE-NAME)
             (COND
                ((SETQ DESCRIBERS (GET-DESCRIBERS TYPE-NAME))
                 (DESCRIBE-USING-DESCRIBERS OBJECT (1+ DEPTH)
                        DESCRIBERS))
                ((AND (SETQ INSPECT-MACRO (GET-INSPECT-MACRO OBJECT))
                      (INSPECT-MACRO-USABLE-BY-DESCRIBE? INSPECT-MACRO))
                 (DESCRIBE-USING-INSPECT-MACRO OBJECT (1+ DEPTH)
                        INSPECT-MACRO))
                ((SETQ SUPER-DESCRIBERS (GET-SUPER-DESCRIBERS TYPE-NAME))
                 (DESCRIBE-USING-DESCRIBERS OBJECT (1+ DEPTH)
                        SUPER-DESCRIBERS))
                ((SETQ IL:DECL (OR (IL:FINDRECDECL OBJECT)
                                   (IL:FINDSYSRECDECL OBJECT)))
                 (DESCRIBE-USING-RECORD-DECL OBJECT IL:DECL (1+ DEPTH)))
                (T ;; Punt to printing
                   (PRIN1 OBJECT))))
       (PRIN1 OBJECT)))

(DEFUN DESCRIBE-NEW-LINE (DEPTH)
   (FRESH-LINE)
   (DOTIMES (N (* DEPTH CL::*DESCRIBE-INDENT*))
       (WRITE-CHAR #\Space)))

(DEFUN DESCRIBE-USING-DESCRIBERS (OBJECT DEPTH DESCRIBERS)
   (MAPC #'(LAMBDA (DESCRIBER)
             (IF (AND (CONSP DESCRIBER)
                      (STRINGP (FIRST DESCRIBER)))
                 (MULTIPLE-VALUE-BIND (FIELD EMPTY?)
                     (FUNCALL (SECOND DESCRIBER)
                              OBJECT)
                   (UNLESS EMPTY?
```

```
                            (DESCRIBE-NEW-LINE DEPTH)
                            (FORMAT T "~A: " (FIRST DESCRIBER))
                            (DESCRIBE-INTERNAL FIELD (1+ DEPTH))))
                      (FUNCALL DESCRIBER OBJECT DEPTH)))
          DESCRIBERS))
```

```
(DEFUN GET-SUPER-DESCRIBERS (TYPE)
   ;; Search up super-types of TYPE for describers
   (DO* ((TYPE TYPE (IL:GETSUPERTYPE TYPE))
         (DESCRIBER NIL (GET-DESCRIBERS TYPE)))
        ((OR DESCRIBER (NULL TYPE))
         DESCRIBER)))
```

```
(DEFUN GET-INSPECT-MACRO (OBJECT)
   ;; Search IL:INSPECTMACROS for an inspect macro for OBJECT
   (DECLARE (XCL:GLOBAL IL:INSPECTMACROS))
   (DO* ((TAIL IL:INSPECTMACROS (REST TAIL))
         (HEAD NIL (FIRST TAIL))
         (TYPE NIL (FIRST HEAD))
         (MACRO NIL (TYPECASE TYPE
                      (CONS (AND (EQ (FIRST TYPE)
                                     'IL:FUNCTION)
                                 (FUNCALL (SECOND TYPE)
                                          OBJECT)))
                      (OTHERWISE (TYPEP OBJECT TYPE)))))
        ((OR MACRO (NULL TAIL))
         HEAD)))
```

```
(DEFUN INSPECT-MACRO-USABLE-BY-DESCRIBE? (MACRO)
   (CONSP (REST MACRO)))
```

```
(DEFUN DESCRIBE-USING-INSPECT-MACRO (OBJECT DEPTH MACRO)
   (LET ((FETCHFN (THIRD MACRO))
         (FIELDS (SECOND MACRO)))
        (MAPCAR #'(LAMBDA (FIELD-NAME)
                    (DESCRIBE-NEW-LINE DEPTH)
                    (PRINC FIELD-NAME)
                    (PRINC ": ")
                    (DESCRIBE-INTERNAL (FUNCALL FETCHFN OBJECT FIELD-NAME)
                                       (1+ DEPTH)))
                (IF (CONSP FIELDS)
                    FIELDS
                    (FUNCALL FIELDS OBJECT)))))
```

```
(DEFUN DESCRIBE-USING-RECORD-DECL (OBJECT IL:DECL DEPTH)
   (MAPC #'(LAMBDA (FIELD-NAME)
             (DESCRIBE-NEW-LINE DEPTH)
             (FORMAT T "~A: " FIELD-NAME)
             (DESCRIBE-INTERNAL (IL:RECORDACCESS FIELD-NAME OBJECT IL:DECL)
                                (1+ DEPTH)))
         (IL:INSPECTABLEFIELDNAMES IL:DECL)))
```

```
(DEFUN A-OR-AN (WORD)
   "Return 'a' or 'an' depending upon whether the first letter in WORD is a vowel"
   (IF (VOWEL-P (ELT (ETYPECASE WORD
                       (SYMBOL (SYMBOL-NAME WORD))
                       (STRING WORD))
                     0))
       "an"
       "a"))
```

```
(DEFUN VOWEL-P (CHAR)
   "T if char is an A, E, I, O or U.  Not dependable with funky charsets."
   (CASE (CHARACTER CHAR)
       ((#\A #\a #\E #\e #\I #\i #\O #\o #\U #\u) T)
       (OTHERWISE NIL)))
```

```
(XCL:DEF-DEFINE-TYPE CL::DESCRIBERS "Describers of objects")
```

```
(XCL:DEFDEFINER CL::DEFDESCRIBER CL::DESCRIBERS (TYPE &REST CL::DESCRIBERS)
   `(SETF (GET ',TYPE 'DESCRIBERS)
          (LIST ,@(MAPCAR #'(LAMBDA (CL::ITEM)
                              ;; Throughout here symbols are quoted and lambda-expressions are hash-quoted for compiler
                              (IF (AND (CONSP CL::ITEM)
                                       (STRINGP (FIRST CL::ITEM)))
                                  ;; It's a field name and function
```

```
                                        `(LIST ',(FIRST CL::ITEM)
                                               (IF (CONSP ',(SECOND CL::ITEM))
                                                   #',(SECOND CL::ITEM)
                                                   ',(SECOND CL::ITEM)))
                                        ;; Else, it must be just a function
                                        (IF (CONSP CL::ITEM)
                                            `#',CL::ITEM
                                            `',CL::ITEM)))
                              CL::DESCRIBERS))))

(DEFUN GET-DESCRIBERS (TYPE)
   (GET TYPE 'DESCRIBERS))

(IL:PUTPROPS DESCRIBERS IL:PROPTYPE IGNORE)


(CL::DEFDESCRIBER SYMBOL
   ;; This describer uses all features
   ("name" SYMBOL-NAME)                                          ; A field name and accessor
   (LAMBDA (SYMBOL CL::DEPTH)                                    ; An arbitrary function
          (LET ((CL::FIRST-TIME? 'T)
                (CL::HASH-TABLES))
               (MAPHASH #'(LAMBDA (TYPE HASH-TABLE)
                                  (WHEN (NOT (MEMBER HASH-TABLE CL::HASH-TABLES :TEST #'EQ))
                                      (PUSH HASH-TABLE CL::HASH-TABLES)
                                      (LET ((CL::DOC (GETHASH SYMBOL HASH-TABLE)))
                                           (WHEN CL::DOC
                                               (WHEN CL::FIRST-TIME?
                                                   (SETQ CL::FIRST-TIME? 'NIL)
                                                   (DESCRIBE-NEW-LINE CL::DEPTH)
                                                   (PRINC "documentation:"))
                                               (DESCRIBE-NEW-LINE (1+ CL::DEPTH))
                                               (FORMAT T "~A: ~A" TYPE CL::DOC)
                                               NIL))))
                        IL:*DOCUMENTATION-HASH-TABLE*)))
   ("package cell" SYMBOL-PACKAGE)                               ; another field name & accessor
   ("value cell"                                                 ; use of multiple values in accessor
          (LAMBDA (SYMBOL)
                 (LET ((CL::UNBOUND? (NOT (BOUNDP SYMBOL))))
                      (VALUES (UNLESS CL::UNBOUND? (SYMBOL-VALUE SYMBOL))
                              CL::UNBOUND?))))
   ("function cell"                                              ; ditto
          (LAMBDA (SYMBOL)
                 (LET ((CL::UNDEFINED? (NOT (FBOUNDP SYMBOL))))
                      (VALUES (UNLESS CL::UNDEFINED? (SYMBOL-FUNCTION SYMBOL))
                              CL::UNDEFINED?))))
   (LAMBDA (SYMBOL CL::DEPTH)                                    ; arbitratry function again
          (LET ((CL::PLIST (SYMBOL-PLIST SYMBOL)))
               (WHEN CL::PLIST
                   (DESCRIBE-NEW-LINE CL::DEPTH)
                   (PRINC "property list:")
                   (DO ((CL::PLIST CL::PLIST (CDDR CL::PLIST)))
                       ((NULL CL::PLIST))
                     (DESCRIBE-NEW-LINE (1+ CL::DEPTH))
                     (PRIN1 (FIRST CL::PLIST))
                     (PRINC " : ")
                     ;; Recurse on each property
                     (DESCRIBE-INTERNAL (SECOND CL::PLIST)
                            (+ CL::DEPTH 2)))))))

(CL::DEFDESCRIBER CL::STRUCTURE-OBJECT
   ;; Describer for objects created by DEFSTRUCT
   (LAMBDA (CL::OBJECT CL::DEPTH)
          (MAPC #'(LAMBDA (CL::SLOT)
                        (DESCRIBE-NEW-LINE CL::DEPTH)
                        (FORMAT T "~A: " (CL::PSLOT-NAME CL::SLOT))
                        ;; Recurse on fields
                        (DESCRIBE-INTERNAL (FUNCALL (CL::PSLOT-ACCESSOR CL::SLOT)
                                                    CL::OBJECT)
                                           (1+ CL::DEPTH)))
                (CL::PS-ALL-SLOTS (CL::PARSED-STRUCTURE (TYPE-OF CL::OBJECT))))))

(CL::DEFDESCRIBER CHARACTER
   (LAMBDA (CHAR CL::DEPTH)
          (MULTIPLE-VALUE-CALL 'FORMAT T "'~:@C', code #\\~O-~3,'0O (~D decimal, ~:*~X hex, ~:*~B binary)" CHAR
                 (FLOOR (CHAR-CODE CHAR)
                        256)
                 (CHAR-CODE CHAR))))
```

```
(CL::DEFDESCRIBER FIXNUM (LAMBDA (NUMBER CL::DEPTH)
                           (FORMAT T "~D decimal, ~:*~O octal, ~:*~X hex, ~:*~B binary~@[, '~C'
                                  character~]" NUMBER (INT-CHAR NUMBER))))

(CL::DEFDESCRIBER SINGLE-FLOAT
   ("sign" (LAMBDA (FLOAT)
             (ECASE (FLOAT-SIGN FLOAT)
                (1.0 'CL::POSITIVE)
                (-1.0 'CL::NEGATIVE))))
   ("radix" FLOAT-RADIX)
   ("digits" FLOAT-DIGITS)
   ("significand" (LAMBDA (FLOAT)
                    ;; onlyt return first value, as second confuses describe.
                    (VALUES (DECODE-FLOAT FLOAT))))
   ("exponent" (LAMBDA (FLOAT)
                 (SECOND (MULTIPLE-VALUE-LIST (DECODE-FLOAT FLOAT))))))

(CL::DEFDESCRIBER HASH-TABLE ("count" HASH-TABLE-COUNT)
                             ("size" IL:HARRAYSIZE)
                             ("test" (LAMBDA (CL::TABLE)
                                       (IL:HARRAYPROP CL::TABLE 'IL:EQUIVFN)))
                             (LAMBDA (CL::TABLE CL::DEPTH)
                                    (DESCRIBE-NEW-LINE CL::DEPTH)
                                    (PRINC "contents:")
                                    (LET* ((CL::NEW-DEPTH (1+ CL::DEPTH))
                                           (CL::NEW-NEW-DEPTH (1+ CL::NEW-DEPTH)))
                                          (MAPHASH #'(LAMBDA (CL::KEY CL::VALUE)
                                                             (DESCRIBE-NEW-LINE CL::NEW-DEPTH)
                                                             (PRIN1 CL::KEY)
                                                             (PRINC " : ")
                                                             (DESCRIBE-INTERNAL CL::VALUE CL::NEW-NEW-DEPTH))
                                                   CL::TABLE))))

(DEFPARAMETER CL::*DESCRIBE-DEPTH* 1
   "The recursive depth to which DESCRIBE describes")

(DEFPARAMETER CL::*DESCRIBE-INDENT* 3
   "Number of spaces to indent recursive descriptions")

(DEFPARAMETER CL::*DESCRIBE-PRINT-LENGTH* 3
   "The value of *PRINT-LENGTH* in DESCRIBE")

(DEFPARAMETER CL::*DESCRIBE-PRINT-LEVEL* 3
   "The value of *PRINT-LEVEL* in DESCRIBE")

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

(IL:PUTPROPS IL:DESCRIBE IL:FILETYPE :XCL-COMPILE-FILE)

(IL:PUTPROPS IL:DESCRIBE IL:COPYRIGHT ("Venue & Xerox Corporation" 1986 1987 1988 1990))
```

FUNCTION INDEX

DESCRIBER INDEX

VARIABLE INDEX

PROPERTY INDEX

DEFINE-TYPE INDEX

DEFINER INDEX