File created: 28-Apr-2022 08:52:36 **{DSK}<Users>kaplan>Local>medley3.5>my-medley>sources>LLCHAR.;13**

changes to: (I.S.OPRS inpname)

previous date: 23-Apr-2022 17:19:02 {DSK}<Users>kaplan>Local>medley3.5>my-medley>sources>LLCHAR.;12

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ LLCHARCOMS
       ((FNS ALLOCSTRING MKATOM SUBATOM CHARACTER \PARSE.NUMBER \INVALID.DOTTED.SYMBOL \INVALID.INTEGER
             \MKINTEGER MKSTRING \PRINDATUM.TO.STRING BKSYSBUF NCHARS NTHCHARCODE RPLCHARCODE \RPLCHARCODE
             NTHCHAR RPLSTRING SUBSTRING GNC GNCCODE GLC GLCCODE STREQUAL STRING.EQUAL STRINGP CHCON1 U-CASE
             L-CASE U-CASEP \SMASHABLESTRING \MAKEWRITABLESTRING \SMASHSTRING \FATTENSTRING)
        (COMS                                                   ; Temporary until low level system is changed to call
                                                                ; STRING.EQUAL again
              (P (MOVD? 'STRING.EQUAL 'STRING-EQUAL NIL T)
                 (MOVD? 'STRING.EQUAL 'CL::SIMPLE-STRING-EQUAL NIL T)))
        (FNS \GETBASESTRING \PUTBASESTRING \PUTBASESTRINGFAT GetBcplString SetBcplString)
        (DECLARE%: DONTCOPY (EXPORT (RECORDS STRINGP)
                                    (GLOBALVARS \OneCharAtomBase)
                                    (RESOURCES \NUMSTR \NUMSTR1 \PNAMESTRING)
                                    (CONSTANTS (\FATPNAMESTRINGP T))
                                    (MACROS \PNAMESTRINGPUTCHAR)
                                    (OPTIMIZERS FCHARACTER)
                                    ;; Iterators expose control variables, $$OFFSET corresponds to current character (except inside user's
                                    ;; repeatwhile or repeatuntil)
                                    (I.S.OPRS inpname inatom instring)
                                                                ; For use when the inner-loop test in the generic operators is too
                                                                ; expensive
                                    (I.S.OPRS infatatom inthinatom infatstring inthinstring)
                                    (MACROS \CHARCODEP \FATCHARCODEP \THINCHARCODEP)
                                                                ; For benefit of Masterscope
                                    (MACROS \GETBASEFAT \GETBASETHIN \PUTBASEFAT \PUTBASETHIN)
                                    (MACROS \PUTBASECHAR \GETBASECHAR)
                                    (MACROS \CHARSET \CHAR8CODE)
                                    (CONSTANTS (\CHARMASK 255)
                                           (\MAXCHAR 255)
                                           (\MAXTHINCHAR 255)
                                           (\MAXFATCHAR 65535)
                                           (\MAXCHARSET 255)
                                           (%#STRINGPWORDS 4))
                                    (MACROS \NATOMCHARS \NSTRINGCHARS)))
        (INITRESOURCES \NUMSTR \NUMSTR1 \PNAMESTRING)
        (P (MOVD? 'CHARACTER 'FCHARACTER NIL T))
        [COMS (FNS %%COPY-ONED-ARRAY %%COPY-STRING-TO-ARRAY)
                                                                ; For MAKEINIT
              (DECLARE%: DONTCOPY (ADDVARS (INEWCOMS (FNS ALLOCSTRING %%COPY-ONED-ARRAY %%COPY-STRING-TO-ARRAY))
                                           (* "So %%COPY-ONED-ARRAY will compile properly")
                                           (INEWCOMS (FILES (SYSLOAD FROM VALUEOF DIRECTORIES)
                                                            CMLARRAY-SUPPORT))
                                           (EXPANDMACROFNS \PUTBASETHIN \PUTBASEFAT \CHARCODEP \GETBASECHAR
                                                  \GETBASETHIN \GETBASEFAT \PUTBASECHAR)
                                           (DONTCOMPILEFNS %%COPY-ONED-ARRAY %%COPY-STRING-TO-ARRAY]
        (DECLARE%: DONTCOPY EVAL@COMPILE (LOCALVARS . T))
        ;; Arrange for the proper compiler
        (PROP FILETYPE LLCHAR)))

(DEFINEQ

(ALLOCSTRING
  [LAMBDA (N INITCHAR OLD FATFLG)                               (* jop%: "23-Sep-86 17:44")
    (SETQ N (FIX N))                                            ; Coerce floats at the outset
    (COND
       ((OR (ILESSP N 0)
            (IGREATERP N \MaxArrayLen))
        (LISPERROR "ILLEGAL ARG" N)))
    [COND
       ((NULL INITCHAR)
        (SETQ INITCHAR 0))
       ((\CHARCODEP INITCHAR))
       (T (SETQ INITCHAR (CHCON1 INITCHAR]
    [LET ((FATP (OR FATFLG (IGREATERP INITCHAR \MAXTHINCHAR)))
          STRINGBASE)                                           ; Allocate the block before going uninterruptable in the smashing
                                                                ; case.
         [SETQ STRINGBASE (\ALLOCBLOCK (COND
                                       (FATP (FOLDHI N WORDSPERCELL))
                                       (T (FOLDHI N BYTESPERCELL]
```

```
         [COND
            [(STRINGP OLD)
             (UNINTERRUPTABLY
                  (create STRINGP smashing OLD LENGTH _ N BASE _ STRINGBASE TYP _ (COND
                                                                                     (FATP \ST.POS16)
                                                                                     (T \ST.BYTE))))]
            (T (SETQ OLD (create STRINGP
                                 LENGTH _ N
                                 BASE _ STRINGBASE
                                 TYP _ (COND
                                          (FATP \ST.POS16)
                                          (T \ST.BYTE]
         (COND
            ((NEQ 0 INITCHAR)                                    ; \ALLOCBLOCK always zeros the block, so don't need to
                                                                 ; initialize then
             (COND
                (FATP (for I from 0 to (SUB1 N) do (\PUTBASEFAT STRINGBASE I INITCHAR)))
                (T (for I from 0 to (SUB1 N) do (\PUTBASETHIN STRINGBASE I INITCHAR]
    OLD])
```

(MKATOM

```
  [LAMBDA (X)                                                    (* jop%: "23-Sep-86 16:30")
    (COND
       ((STRINGP X)
        (\MKATOM (ffetch (STRINGP BASE) of X)
               (ffetch (STRINGP OFFST) of X)
               (LET ((LEN (ffetch (STRINGP LENGTH) of X)))
                    (COND
                       ((IGREATERP LEN \PNAMELIMIT)
                        (LISPERROR "ATOM TOO LONG" X))
                       (T LEN)))
               (ffetch (STRINGP FATSTRINGP) of X)))
       ((OR (LITATOM X)
            (NUMBERP X))
        X)
       (T (PACK* X])
```

(SUBATOM

```
  [LAMBDA (X N M)                                                (* jop%: "23-Sep-86 17:47")
    (PROG (BASE OFFST LEN FATP (N1 N)
                (M1 M))                                          ; N1 and M1 so don't reset user arg.
          [COND
             ((LITATOM X)
              (SETQ BASE (ffetch (LITATOM PNAMEBASE) of X))
              (SETQ OFFST 1)
              (SETQ FATP (ffetch (LITATOM FATPNAMEP) of X))
              (SETQ LEN (ffetch (LITATOM PNAMELENGTH) of X)))
             (T (SETQ LEN (OR (STRINGP X)
                              (MKSTRING X)))                     ; Don't reset user arg
                (SETQ BASE (ffetch (STRINGP BASE) of LEN))
                (SETQ FATP (ffetch (STRINGP FATSTRINGP) of LEN))
                (SETQ OFFST (ffetch (STRINGP OFFST) of LEN))
                (SETQ LEN (ffetch (STRINGP LENGTH) of LEN]
          [COND
             ((IGREATERP 0 N1)                                   ; Coerce the first index
              (SETQ N1 (IPLUS N1 LEN 1]
          [COND
             ((NULL M1)                                          ; Coerce the second
              (SETQ M1 LEN))
             ((IGREATERP 0 M1)
              (SETQ M1 (IPLUS M1 LEN 1]
          (RETURN (AND (IGREATERP N1 0)
                       (ILEQ N1 M1)
                       (ILEQ M1 LEN)
                       (\MKATOM BASE (IPLUS OFFST N1 -1)
                              (COND
                                 ((IGREATERP (SETQ LEN (ADD1 (IDIFFERENCE M1 N1)))
                                         \PNAMELIMIT)
                                  (LISPERROR "ATOM TOO LONG" X))
                                 (T LEN))
                              FATP])
```

(CHARACTER

```
  [LAMBDA (N)                                                    (* jop%: "23-Sep-86 17:45")
    (OR (\CHARCODEP N)
        (SETQ N (\ILLEGAL.ARG N)))
    (COND
       ((IGREATERP N \MAXTHINCHAR)                               ; The character we're getting is NOT a thin character -- do it the
                                                                 ; hard way
        (WITH-RESOURCE (\PNAMESTRING)
               (\PNAMESTRINGPUTCHAR (ffetch (STRINGP XBASE) of \PNAMESTRING)
                      0 N)
               (\MKATOM (ffetch (STRINGP XBASE) of \PNAMESTRING)
                      0 1 \FATPNAMESTRINGP)))
```

```
       ((IGREATERP N (CHARCODE 9))
        (\ADDBASE \OneCharAtomBase (IDIFFERENCE N 10)))
       ((IGEQ N (CHARCODE 0))
        (IDIFFERENCE N (CHARCODE 0)))
       (T                                                ; The common case -- just add on the one-atom base.
          (\ADDBASE \OneCharAtomBase N])
```

(\PARSE.NUMBER

```
  [LAMBDA (BASE BN LEN FATP RADIX RDTBL)                ; Edited 12-Feb-87 19:21 by bvm:
```

;;; Attempt to create a numeric atom out of the chars in BASE from BN for LEN characters (fat or thin, depending on FATP). Return NIL if the chars do ;;; not form a legal number when read in this read table.

```
    (DECLARE (GLOBALVARS \ORIGREADTABLE))
    (if (NULL RDTBL)
        then (SETQ RDTBL *READTABLE*))
    (PROG ((I BN)
           (END (IPLUS BN LEN))
           (STATE 'INIT)
           (COMMONLISP (AND (NEQ RDTBL \ORIGREADTABLE)
                            (fetch (READTABLEP COMMONLISP) of RDTBL)))
           COMMONLISPY MAXDIGIT MAXALPHADIGIT C SIGN START ENDFRAC DECPT EXPSTART NEGFRAC SIGDIGITS EXP10
           SEENALPHADIGITS SEENBOGUSDIGITS)
                                                        ; The test for \origreadtable is a kludge so that \MKATOM can
                                                        ; work before read tables are set up.  \MKATOM calls us with
                                                        ; RDTBL = \origreadtable, which is initially NOBIND.
          (if (NULL RADIX)
              then (SETQ RADIX (if COMMONLISP
                                   then *READ-BASE*
                                 else 10)))
          [if (GREATERP RADIX 10)
              then                                      ; can have alphabetic digits for large bases
                   (SETQ MAXALPHADIGIT (IPLUS (CHARCODE A)
                                              (IDIFFERENCE RADIX 11)))
                   (SETQ MAXDIGIT (CHARCODE 9))
            else (SETQ MAXDIGIT (IPLUS (CHARCODE 0)
                                       (SUB1 RADIX]
          [SETQ COMMONLISPY (OR COMMONLISP (AND (NEQ RDTBL \ORIGREADTABLE)
                                                (fetch (READTABLEP COMMONNUMSYNTAX) of RDTBL]
      LP
          ;; Scan string to see what we have: a decimal integer, octal integer, or floating-point number.  Once we know which we have, we can pack up the
          ;; value quickly
          (if (EQ I END)
              then (RETURN (SELECTQ STATE
                                    ((INITDIGIT AFTERQ AFTERMIDDLEDOT)
                                         (if (NOT START)
                                             then                   ; saw no non-zero digits
                                                  0
                                           elseif SEENBOGUSDIGITS
                                             then                   ; Some digits were not valid in this radix, so object is not a
                                                                    ; number.  Note that there is no suffix in this case, so i is correct.
                                                  (\INVALID.INTEGER BASE START I SIGN RADIX FATP)
                                           else (\MKINTEGER BASE START (if (NEQ STATE 'INITDIGIT)
                                                                           then
                                                                    ; string ended in Q or dot
                                                                           (SUB1 I)
                                                                         else I)
                                                            (EQ SIGN '-)
                                                            RADIX FATP)))
                                    ((INFRACTION INEXPONENT)
                                         (if SIGDIGITS
                                             then [if (NOT ENDFRAC)
                                                      then (SETQ ENDFRAC I)
                                                           (SETQ NEGFRAC (EQ SIGN '-]
                                                  (if (IGREATERP SIGDIGITS MAX.DIGITS.ACCURACY)
                                                      then
                                                  ;; Too many digits--we will overflow.  Only take as many as we can handle.  Don't worry about looking
                                                  ;; at the n+1'st digit for rounding, since it won't make any difference (there are many fewer sig bits in a
                                                  ;; floatp than in a fixp)
                                                           (SETQ ENDFRAC (IPLUS START MAX.DIGITS.ACCURACY))
                                                           (if (AND (IGREATERP DECPT START)
                                                                    (ILESSP DECPT ENDFRAC))
                                                               then (add ENDFRAC 1)))
                                                  (SETQ EXP10 (if EXPSTART
                                                                  then (\MKINTEGER BASE EXPSTART I (EQ SIGN
                                                                                                       '-)
                                                                                   10 FATP)
                                                                else 0))
                                                                    ; the explicit exponent
                                                  (\FLOATINGSCALE (\MKINTEGER BASE START ENDFRAC NEGFRAC 10 FATP)
                                                         (IPLUS EXP10 (IDIFFERENCE DECPT ENDFRAC)
                                                                (if (ILESSP DECPT ENDFRAC)
                                                                    then
                                                                    ; don't count the position the dec pt occupies
                                                                      1
```

```
                                                    else 0)))
                                    else                        ; we saw only zeros
                                         (FLOAT 0)))
                                NIL)))
          (SETQ STATE
           (OR
            [SELCHARQ (SETQ C (\GETBASECHAR FATP BASE I))
                 (- (AND (NOT SIGN)
                         (SELECTQ STATE
                              ((INIT AFTERE)
                                   (SETQ SIGN '-)
                                   STATE)
                              NIL)))
                 (+ (AND (NOT SIGN)
                         (SELECTQ STATE
                              ((INIT AFTERE)
                                   (SETQ SIGN '+)
                                   STATE)
                              NIL)))
                 (%. (SETQ DECPT I)
                     (SELECTQ STATE
                          (INIT 'AFTERINITIALDOT)
                          (INITDIGIT (if SEENALPHADIGITS
                                         then                   ; Can't have decimal point in other radices
                                              NIL
                                       elseif COMMONLISP
                                         then                   ; Could be decimal integer
                                              (SETQ RADIX 10)
                                              (SETQ SEENBOGUSDIGITS NIL)
                                                                ; digits bigger than radix not an error any more
                                              'AFTERMIDDLEDOT
                                       else 'INFRACTION))
                          (AFTERINITIALDOT                      ; Two dots in a row.  If symbol is ALL dots, then we have to
                                                                ; signal an error.
                               (if [AND COMMONLISP (NOT SIGN)
                                        (for J from (ADD1 I) to (SUB1 END)
                                           always (EQ (\GETBASECHAR FATP BASE J)
                                                      (CHARCODE %.]
                                   then (\INVALID.DOTTED.SYMBOL BASE BN LEN FATP)
                                 else                           ; not all dots, started with sign, or in Interlisp read table, where
                                                                ; it's ok -- just not a number
                                      NIL))
                          NIL))
                 (COND
                    ((AND (IGEQ C (CHARCODE 0))
                          (ILEQ C (CHARCODE 9)))                ; digit
                     (SELECTQ STATE
                          ((INIT INITDIGIT)
                               (IF (> C MAXDIGIT)
                                   THEN                         ; not a digit in this radix.  However, number could turn out to be
                                                                ; decimal (integer or float), so keep going.
                                       (SETQ SEENBOGUSDIGITS T))
                               (if SIGDIGITS
                                   then (add SIGDIGITS 1)
                                 elseif (NEQ C (CHARCODE 0))
                                   then                         ; record where first significant digit happens
                                        (SETQ START I)
                                        (SETQ SIGDIGITS 1))
                               'INITDIGIT)
                          ((INFRACTION AFTERINITIALDOT AFTERMIDDLEDOT)
                                                                ; Scanning fractional part
                               (if SIGDIGITS
                                   then (add SIGDIGITS 1)
                                 elseif (NEQ C (CHARCODE 0))
                                   then (SETQ SIGDIGITS 1)
                                        (SETQ START I))
                               'INFRACTION)
                          (AFTERE (SETQ EXPSTART I)
                                  'INEXPONENT)
                          (INEXPONENT 'INEXPONENT)
                          NIL))
                    ((IGREATERP C (CHARCODE z))                 ; Out in the wilderness
                     NIL)
                    (T                                          ; Some other non-digit
                     [if (AND COMMONLISPY (IGEQ C (CHARCODE a)))
                         then (SETQ C (IDIFFERENCE C (IDIFFERENCE (CHARCODE a)
                                                                  (CHARCODE A]
                     (if (AND MAXALPHADIGIT (IGEQ C (CHARCODE A))
                              (ILEQ C MAXALPHADIGIT)
                              (NOT DECPT))
                         then                                   ; Letter is a digit in this base
                              (SELECTQ STATE
                                   ((INIT INITDIGIT)
                                        (SETQ SEENALPHADIGITS T)
                                        (if SIGDIGITS
                                            then (add SIGDIGITS 1)
                                          else (SETQ START I)
```

```
                                        (SETQ SIGDIGITS 1))
                                    'INITDIGIT)
                                NIL)
                           elseif (EQ C (CHARCODE Q))
                             then                                    ; Interlisp octal specifier -- perhaps should only do this if not
                                                                     ; common lisp
                                  (SELECTQ STATE
                                       (INITDIGIT (SETQ RADIX 8)
                                                  (SETQ SEENBOGUSDIGITS NIL)
                                                                     ; It is possible that we should check to see if all the digits are
                                                                     ; really octal digits, but that's a pain, and we never did it before in
                                                                     ; Interlisp.
                                                  'AFTERQ)
                                       NIL)
                           elseif (AND [OR (EQ C (CHARCODE E))
                                           (AND COMMONLISPY (FMEMB C (CHARCODE (D F L S]
                                       (NOT SEENALPHADIGITS))
                             then                                    ; Exponent marker.  Someday there will be differences among
                                                                     ; some of these
                                  (SELECTQ STATE
                                       ((INITDIGIT INFRACTION AFTERMIDDLEDOT)
                                                                     ; We've seen digits and/or a fraction
                                            (OR DECPT (SETQ DECPT I))
                                            (SETQ ENDFRAC I)
                                            (SETQ NEGFRAC (EQ SIGN '-))
                                            (SETQ SIGN NIL)
                                            'AFTERE)
                                       NIL)
                           elseif (AND (EQ C (CHARCODE /))
                                       COMMONLISPY)
                             then                                    ; Ratio marker.  Must only have seen digits and possibly sign so
                                                                     ; far
                              (if
                               [AND (EQ STATE 'INITDIGIT)
                                    (NEQ (ADD1 I)
                                         END)
                                    (for J from (ADD1 I) to (SUB1 END)
                                       always                        ; test remaining digits valid for this radix
                                              (AND (IGEQ (SETQ C (\GETBASECHAR FATP BASE J))
                                                         (CHARCODE 0))
                                                   (OR (ILEQ C MAXDIGIT)
                                                       (AND MAXALPHADIGIT (IGEQ C (CHARCODE A))
                                                            (ILEQ (if (IGEQ C (CHARCODE a))
                                                                      then (IDIFFERENCE C (IDIFFERENCE (CHARCODE a)
                                                                                                       (CHARCODE A)))
                                                                      else C)
                                                                  MAXALPHADIGIT]
                                  then (RETURN (if START
                                                   then (/ (\MKINTEGER BASE START I (EQ SIGN '-)
                                                                       RADIX FATP)
                                                           (\MKINTEGER BASE (ADD1 I)
                                                                       END NIL RADIX FATP))
                                                else                 ; saw no non-zero digits
                                                     0]
                    (RETURN NIL)))
                  (SETQ I (ADD1 I))
                  (GO LP])
```

(**INVALID.DOTTED.SYMBOL**

```
  [LAMBDA (BASE START LEN FATP)                                      ; Edited 12-Feb-87 18:56 by bvm:
```

;;; Called from number parser when scanning a token that is all dots. Value returned from here is NIL to treat it as a quoted symbol or any other non-null
;;; value you'd like to return.

```
    (CL:CERROR "Treat the dots as if they were escaped" "Invalid symbol consisting entirely of dots ~S"
           (\GETBASESTRING BASE START LEN FATP))
    NIL])
```

(**INVALID.INTEGER**

```
  [LAMBDA (BASE START END SIGN RADIX FATP)                           ; Edited 12-Feb-87 19:39 by bvm:
```

;;; Called when scanning a token that is all digits, but some digits are not valid in this read base. Value returned from here is NIL to treat it as a symbol
;;; or a number (the default proceed case says to interpret in decimal).

```
    (CL:CERROR "Treat the number as if in decimal radix" "Invalid integer %"~@[~A~]~A%" in read base ~D" SIGN
           (\GETBASESTRING BASE (if FATP
                                    then
                                     ;; yecch.  start arg to \getbasestring is always byte offset, whether it's fat or not.  start arg to
                                     ;; \parse.number is character number (and usually zero, apparently).
                                        (UNFOLD START BYTESPERWORD)
                                    else START)
                  (- END START)
                  FATP)
           RADIX)
```

```
    (\MKINTEGER BASE START END (EQ SIGN '-)
           10 FATP])
```

(\MKINTEGER

```
  [LAMBDA (BASE START END NEG RADIX FATP)                    ; Edited 13-Oct-87 11:10 by jrb:
```

;;; Return integer whose Ascii characters run from START to END off BASE. If NEG is true, negate it. RADIX is the base. String is assumed to contain
;;; only digits valid in RADIX -- no error checking. For benefit of floating routines, dec pt is ignored.

;;; JRB - Modified per BvM suggestion to accumulate three digits at a time (three digits insures largest legal radix (36) won't overflow a smallp). The
;;; bottom of the loop goes to great lengths to avoid computing RADIX^2 and RADIX^3 unless it absolutely has to.

```
    (PROG ((VAL 0)
           LOOPVAL CH I RADIX2 RADIX3)
      LP   (if (EQ START END)
               then (RETURN VAL))
           (SETQ LOOPVAL 0)
           (SETQ I 3)
           (while (AND (NOT (EQ START END))
                       (NOT (EQ I 0)))
              do (SETQ CH (\GETBASECHAR FATP BASE START))
                 (if (NEQ CH (CHARCODE "."))
                     then                                    ; ignore dec pt
                          [SETQ CH (if (IGEQ CH (CHARCODE A))
                                       then                  ; Large radix digit.  Could be lowercase, so zap the 40q bit
                                            (IPLUS 10 (IDIFFERENCE (LOGAND CH 95)
                                                                   (CHARCODE A)))
                                     else (IDIFFERENCE CH (CHARCODE 0]
                          (SETQ LOOPVAL (if NEG
                                            then (IDIFFERENCE (ITIMES LOOPVAL RADIX)
                                                              CH)
                                          else (IPLUS (ITIMES LOOPVAL RADIX)
                                                      CH)))
                          (SETQ I (SUB1 I)))
                 (SETQ START (ADD1 START)))
           (SETQ VAL (if (EQ VAL 0)
                         then LOOPVAL
                       else [OR RADIX3 (SETQ RADIX3 (ITIMES RADIX (SETQ RADIX2 (ITIMES RADIX RADIX]
                                (IPLUS (ITIMES VAL (SELECTQ I
                                                           (0 RADIX3)
                                                           (1 RADIX2)
                                                           (2 RADIX)
                                                           1))
                                       LOOPVAL)))
           (GO LP])
```

(MKSTRING

```
  [LAMBDA (X FLG RDTBL)                                      ; Edited 10-Feb-87 19:09 by bvm:
                                                             ; Coerce X to be a string.  The string will be FAT if X is
    (DECLARE (GLOBALVARS PRXFLG))
    (OR [COND
          ((NOT FLG)                                         ; The simple case -- just gather up the characters in the item
           (COND
              ((STRINGP X)                                   ; Strings coerce to themselves
               X)
              [(LITATOM X)                                   ; LITATOMs have a new descriptor created, pointing to the same
                                                             ; characters.
               (create STRINGP
                       XBASE _ (ffetch (LITATOM PNAMEBASE) of X)
                       LENGTH _ (ffetch (LITATOM PNAMELENGTH) of X)
                       OFFST _ 1
                       XREADONLY _ T
                       TYP _ (COND
                                ((ffetch (LITATOM FATPNAMEP) of X)
                                 \ST.POS16)
                                (T \ST.BYTE]
              ((CL:CHARACTERP X)                             ; CL characters are one-character strings
               (ALLOCSTRING 1 (CL:CHAR-CODE X]
       (LET [(BASE (COND
                     (PRXFLG (\CHECKRADIX *PRINT-BASE*))
                     (T 10]
            (LET ((*PRINT-ESCAPE* FLG)
                  (*READTABLE* (COND
                                  (FLG (\GTREADTABLE RDTBL))
                                  (T *READTABLE*)))
                  (*PRINT-RADIX* (AND FLG (NEQ BASE 10)))
                  (*PRINT-BASE* BASE)
                  (*PRINT-LENGTH*)
                  (*PRINT-LEVEL*))
                 ;; General case: internally print the name, gather up the characters
                 (\PRINDATUM.TO.STRING X])
```

(\PRINDATUM.TO.STRING

```
[LAMBDA (X)                                                    ; Edited  9-Dec-86 11:04 by jrb:

;;; Produces a string that is the result of printing X according the current settings of *PRINT-ESCAPE* etc.

    (SELECTC (NTYPX X)
        ((LIST \FIXP \SMALLP \FLOATP)                          ; We know how to print numbers without extra steps
            (GLOBALRESOURCE (\NUMSTR \NUMSTR1)
                   (LET [(STR (COND
                                  ((FLOATP X)
                                   (\CONVERT.FLOATING.NUMBER X \NUMSTR \NUMSTR1))
                                  (T (\CONVERTNUMBER X *PRINT-BASE* NIL (AND *PRINT-RADIX* *READTABLE*)
                                                    \NUMSTR \NUMSTR1]
                         (RPLSTRING (ALLOCSTRING (NCHARS STR))
                                    1 STR))))
        (LET ((FATSTRINGP)
              (STRINGLEN 0)
              (STRINDEX 0)
              STRINGPTR *PRINT-CIRCLE-HASHTABLE* (*PRINT-CIRCLE-NUMBER* 1)
              THERE-ARE-CIRCLES)
             (DECLARE (CL:SPECIAL *PRINT-CIRCLE-HASHTABLE* *PRINT-CIRCLE-NUMBER* THERE-ARE-CIRCLES))
                                                               ; If *print-circle* is on, need to scan the structure
             (IF *PRINT-CIRCLE*
                 THEN (SETQ *PRINT-CIRCLE-HASHTABLE* (CL:MAKE-HASH-TABLE))
                      (PRINT-CIRCLE-SCAN X)
                      (IF (NOT THERE-ARE-CIRCLES)
                          THEN (SETQ *PRINT-CIRCLE-HASHTABLE* NIL)))

             ;; First count up the characters and their fatness

             (\MAPPNAME.INTERNAL [FUNCTION (LAMBDA (DUMMY CODE)
                                             (COND
                                                ((GREATERP CODE \MAXTHINCHAR)
                                                 (SETQ FATSTRINGP T)))
                                             (add STRINGLEN 1]
                    X)

             ;; We print structures TWICE here, so we need to reset *PRINT-CIRCLE-HASHTABLE* and *PRINT-CIRCLE-NUMBER* if circles are
             ;; being printed

             (if *PRINT-CIRCLE-HASHTABLE*
                 then (SETQ *PRINT-CIRCLE-NUMBER* 1)
                      (CL:MAPHASH #'[LAMBDA (KEY VAL)
                                      (if (NUMBERP VAL)
                                          then (CL:SETF (CL:GETHASH KEY *PRINT-CIRCLE-HASHTABLE*)
                                                        'T2]
                                  *PRINT-CIRCLE-HASHTABLE*))

             ;; Then print X again actually storing the characters into the string

             (SETQ STRINGPTR (ALLOCSTRING STRINGLEN NIL NIL FATSTRINGP))
             (\MAPPNAME.INTERNAL [FUNCTION (LAMBDA (DUMMY CODE)
                                             [COND
                                                ((EQ STRINDEX (ffetch (STRINGP LENGTH) of STRINGPTR))
                                                                ; Help! NCHARS and \MAPPNAME disagree.
                                                 (SETQ STRINGPTR (CONCAT STRINGPTR " "]
                                             (add STRINDEX 1)
                                             (COND
                                                ((ffetch (STRINGP FATSTRINGP) of STRINGPTR)
                                                                ; Fat string;  just smash the character in.
                                                 (\PUTBASEFAT (fetch (STRINGP BASE) of STRINGPTR)
                                                        (IPLUS (fetch (STRINGP OFFST) of STRINGPTR)
                                                               STRINDEX -1)
                                                        CODE))
                                                ((ILEQ CODE \MAXTHINCHAR)
                                                                ; Thin char and String;  just smash the char in
                                                 (\PUTBASETHIN (fetch (STRINGP BASE) of STRINGPTR)
                                                        (IPLUS (fetch (STRINGP OFFST) of STRINGPTR)
                                                               STRINDEX -1)
                                                        CODE))
                                                (T
                                                 ;; Need to fatten the string, then smash in the char.  This shouldn't happen unless X
                                                 ;; gets printed different the two times!
                                                   (\FATTENSTRING STRINGPTR)
                                                   (\PUTBASEFAT (fetch (STRINGP BASE) of STRINGPTR)
                                                          (IPLUS (fetch (STRINGP OFFST) of STRINGPTR)
                                                                 STRINDEX -1)
                                                          CODE]
                    X)
             STRINGPTR])

(BKSYSBUF
  [LAMBDA (X FLG RDTBL)                                        (* jop%: "23-Sep-86 17:31")
    (PROG NIL
          (if (NOT FLG)
              then (COND
                      [(LITATOM X)
                       (RETURN (for C inatom X do (BKSYSCHARCODE C]
                      [(STRINGP X)
```

```
                    (RETURN (for C instring X do (BKSYSCHARCODE C]
                   (T NIL)))
          (LET ((*READTABLE* *READTABLE*)
                (*PACKAGE* *PACKAGE*)
                TTY)
               [if FLG
                   then (if RDTBL
                            then                                    ; Use the explicit read table we were given
                                 (SETQ *READTABLE* (\GTREADTABLE RDTBL))
                          elseif (NEQ (SETQ TTY (TTY.PROCESS))
                                      (THIS.PROCESS))
                            then                                    ; Print it using the read environment of the destination tty
                                 (SETQ *READTABLE* (PROCESS.EVALV TTY '*READTABLE*))
                                 (SETQ *PACKAGE* (PROCESS.EVALV TTY '*PACKAGE*]
               (\MAPPNAME [FUNCTION (LAMBDA (DUMMY CODE)
                                      (BKSYSCHARCODE CODE]
                     X FLG RDTBL)))
    X])
```

(NCHARS

```
  [LAMBDA (X FLG RDTBL)                                             (* jop%: "24-Sep-86 23:06")
```

;;; Return the number of characters in (the print name of) X. If FLG, then return the number of characters in the PRIN2 version, according to RDTBL.

```
    (PROG ((NCHARCNT 0))
          [COND
             [(LITATOM X)
              (if (NOT FLG)
                  then                                              ; Too hairy to figure out package count
                       (RETURN (ffetch (LITATOM PNAMELENGTH) of X]
             ((STRINGP X)
              (RETURN (IPLUS (ffetch (STRINGP LENGTH) of X)
                             (COND
                                [FLG
                                     ;; 2 for the enclosing quotes and an escape to quote every double quote char or escape in the string
                                     ;; body
                                     (IPLUS 2 (for C instring X bind (ESC _ (ffetch (READTABLEP ESCAPECHAR)
                                                                               of (\GTREADTABLE RDTBL)))
                                                 count (OR (EQ C (CHARCODE %"))
                                                           (EQ C (CHARCODE LF))
                                                           (EQ C ESC]
                                (T 0]                               ; Slow case...
          (\MAPPNAME [FUNCTION (LAMBDA NIL
                                 (add NCHARCNT 1]
                 X FLG RDTBL)
          (RETURN NCHARCNT])
```

(NTHCHARCODE

```
  [LAMBDA (X N FLG RDTBL)                                           (* jop%: "23-Sep-86 16:34")
    (PROG (BASE OFFST FATP LEN (M N))
          [COND
             (FLG (GO SLOWCASE))
             (T (COND
                   ((STRINGP X)
                    (SETQ BASE (ffetch (STRINGP BASE) of X))
                    (SETQ LEN (ffetch (STRINGP LENGTH) of X))
                    (SETQ OFFST (ffetch (STRINGP OFFST) of X))
                    (SETQ FATP (ffetch (STRINGP FATSTRINGP) of X)))
                   ((LITATOM X)
                    (SETQ BASE (ffetch (LITATOM PNAMEBASE) of X))
                    (SETQ LEN (ffetch (LITATOM PNAMELENGTH) of X))
                    (SETQ OFFST 1)
                    (SETQ FATP (ffetch (LITATOM FATPNAMEP) of X)))
                   (T (GO SLOWCASE]
          [COND
             ((ILESSP M 0)                                          ; Negative index counts from end
              (SETQ M (IPLUS M LEN 1]
          [RETURN (COND
                     ((OR (ILESSP M 1)
                          (IGREATERP M LEN))                        ; out of range
                      NIL)
                     (T                                             ; The -1 is cause strings have ORIG=1
                        (\GETBASECHAR FATP BASE (SUB1 (IPLUS OFFST M]
      SLOWCASE
          [COND
             ((EQ M 0)
              (RETURN))
             ((ILESSP M 0)
              (AND (ILESSP (SETQ M (IPLUS M (NCHARS X FLG RDTBL)
                                          1))
                           1)
                   (RETURN]
          (\MAPPNAME [FUNCTION (LAMBDA (DUMMY CODE)
                                 (COND
                                    ((EQ (SETQ M (SUB1 M))
```

```
                                        0)
                                    (RETFROM 'NTHCHARCODE CODE]
                X FLG RDTBL)
          (RETURN])
```

(RPLCHARCODE

```
  [LAMBDA (X N CHAR)                                        (* jop%: "23-Sep-86 16:36")
    (COND
       ((STRINGP X)
        (PROG ((LEN (ffetch (STRINGP LENGTH) of X)))
              (\SMASHABLESTRING X (\FATCHARCODEP CHAR))
              [COND
                 ((ILESSP N 0)                              ; address from end
                  (SETQ N (IPLUS N LEN 1]
              (COND
                 ((OR (ILESSP N 1)
                      (IGREATERP N LEN))
                  (LISPERROR "ILLEGAL ARG" N)))             ; We assume that ORIG is 1 because X is a string
              (\PUTBASECHAR (ffetch (STRINGP FATSTRINGP) of X)
                     (ffetch (STRINGP BASE) of X)
                     (IPLUS (ffetch (STRINGP OFFST) of X)
                            (SUB1 N))
                     CHAR)
              (RETURN X)))
       (T (RPLCHARCODE (MKSTRING X)
                 N CHAR])
```

(\RPLCHARCODE

```
  [LAMBDA (X N CHAR)                                        (* jop%: "23-Sep-86 16:50")
```

;;; System version: does error checking interpreted. Compiles open as \PUTBASEFAT or \PUTBASETHIN. N must be positive, X must be a real not
;;; READONLY string

```
    (COND
       ((OR (NOT (STRINGP X))
            (ffetch (STRINGP READONLY) of X))               ; X has to be a string, and can't be READONLY (e.g. a litatom's
                                                            ; pname)
        (LISPERROR "ILLEGAL ARG" X))
       ((OR (ILEQ N 0)
            (IGREATERP N (ffetch (STRINGP LENGTH) of X)))   ; The position arg has to be inside the string's length
        (LISPERROR "ILLEGAL ARG" N))
       ((NOT (\CHARCODEP CHAR))                             ; CHAR has to be a charcode
        (LISPERROR "ILLEGAL ARG" CHAR))
       ((AND (IGREATERP CHAR \MAXTHINCHAR)
             (NOT (ffetch (STRINGP FATSTRINGP) of X)))      ; If the char's fat, and the string isn't, coerce it to fatness.
        (\SMASHABLESTRING X T)))
    (\PUTBASECHAR (ffetch (STRINGP FATSTRINGP) of X)
           (ffetch (STRINGP BASE) of X)
           (IPLUS (ffetch (STRINGP OFFST) of X)
                  (SUB1 N))
           CHAR)
    X])
```

(NTHCHAR

```
  [LAMBDA (X N FLG RDTBL)                                   (* jop%: "23-Sep-86 17:17")
    (LET ((CODE (NTHCHARCODE X N FLG RDTBL)))
         (AND CODE (FCHARACTER CODE])
```

(RPLSTRING

```
  [LAMBDA (X N Y)                                           ; Edited 24-Sep-87 11:49 by bvm:
    (PROG ((OLDSTRING (OR (STRINGP X)
                          (MKSTRING X)))
           (REP Y)
           OBASE OLEN RBASE RLEN ROFFST POS FIRSTNEW RFAT)
          (SETQ OLEN (ffetch (STRINGP LENGTH) of OLDSTRING))
          [COND
             ((LITATOM REP)
              (SETQ RBASE (ffetch (LITATOM PNAMEBASE) of REP))
              (SETQ ROFFST 1)
              (SETQ RLEN (ffetch (LITATOM PNAMELENGTH) of REP))
              (SETQ RFAT (ffetch (LITATOM FATPNAMEP) of REP)))
             (T (OR (STRINGP REP)
                    (SETQ REP (MKSTRING REP)))
                (SETQ RBASE (ffetch (STRINGP BASE) of REP))
                (SETQ ROFFST (ffetch (STRINGP OFFST) of REP))
                (SETQ RLEN (ffetch (STRINGP LENGTH) of REP))
                (SETQ RFAT (ffetch (STRINGP FATSTRINGP) of REP]
          [COND
             ((> [+ RLEN (SETQ POS (COND
                                      ((> N 0)
                                       (SUB1 N))
                                      (T (+ OLEN N]
                 OLEN)
```

```
            (LISPERROR "ILLEGAL ARG" (if (> POS OLEN)
                                         then                ; actually, the index is wrong, without even considering the
                                                             ; replacement
                                              N
                                       else Y]
         (\SMASHABLESTRING OLDSTRING RFAT)                   ; Make sure the string is writeable and of the appropriate width
         (SETQ OBASE (ffetch (STRINGP BASE) of OLDSTRING))   ; Note: OBASE might have changed, so not fetched until now
         (SETQ FIRSTNEW (+ POS (ffetch (STRINGP OFFST) of OLDSTRING)))
                                                             ; Now can smash chars from RBASE into OBASE starting at
                                                             ; position FIRSTNEW
         (COND
            (RFAT                                            ; Fat into fat. \SMASHABLESTRING* above ensured that
                                                             ; OLDSTRING is now fat
                  (\BLT (\ADDBASE OBASE FIRSTNEW)
                        (\ADDBASE RBASE ROFFST)
                        RLEN))
            [(ffetch (STRINGP FATSTRINGP) of OLDSTRING)      ; Smashing thin string into a fat one
             (for I from ROFFST to (SUB1 (+ ROFFST RLEN)) as J from FIRSTNEW do (\PUTBASEFAT OBASE J
                                                                                (\GETBASETHIN RBASE I]
            (T                                               ; Thin into thin is just byte blt
               (\MOVEBYTES RBASE ROFFST OBASE FIRSTNEW RLEN)))
         (RETURN OLDSTRING])
```

(SUBSTRING

```
  [LAMBDA (X N M OLDPTR)                                     (* jop%: "23-Sep-86 17:48")
    (PROG ((OLDSTRING X)
           (START N)
           (END M)
           FATP BASE OFFST LEN)                              ; OLDSTRING START and END so don't reset user args
          [COND
             ((LITATOM OLDSTRING)
              (SETQ BASE (ffetch (LITATOM PNAMEBASE) of OLDSTRING))
              (SETQ LEN (ffetch (LITATOM PNAMELENGTH) of OLDSTRING))
              (SETQ FATP (ffetch (LITATOM FATPNAMEP) of OLDSTRING))
              (SETQ OFFST 1))
             (T (OR (STRINGP OLDSTRING)
                    (SETQ OLDSTRING (MKSTRING OLDSTRING)))
                (SETQ BASE (ffetch (STRINGP BASE) of OLDSTRING))
                (SETQ LEN (ffetch (STRINGP LENGTH) of OLDSTRING))
                (SETQ FATP (ffetch (STRINGP FATSTRINGP) of OLDSTRING))
                (SETQ OFFST (ffetch (STRINGP OFFST) of OLDSTRING]
          [COND
             ((ILESSP START 0)                               ; Coerce the first index
              (SETQ START (IPLUS START LEN 1]
          [COND
             ((NULL END)                                     ; Now coerce the second index
              (SETQ END LEN))
             ((ILESSP END 0)
              (SETQ END (IPLUS END LEN 1]
          (RETURN (COND
                     ((AND (IGREATERP START 0)
                           (ILEQ START END)
                           (ILEQ END LEN))
                      (UNINTERRUPTABLY
                          [COND
                             ((STRINGP OLDPTR)
                              (create STRINGP smashing OLDPTR READONLY _ (LITATOM OLDSTRING)
                                                       BASE _ BASE TYP _ (COND
                                                                            (FATP \ST.POS16)
                                                                            (T \ST.BYTE))
                                                       LENGTH _ (ADD1 (IDIFFERENCE END START))
                                                       OFFST _ (IPLUS START OFFST -1)))
                             (T (SETQ OLDPTR (create STRINGP
                                                     READONLY _ (LITATOM OLDSTRING)
                                                     BASE _ BASE
                                                     TYP _ (COND
                                                              (FATP \ST.POS16)
                                                              (T \ST.BYTE))
                                                     LENGTH _ (ADD1 (IDIFFERENCE END START))
                                                     OFFST _ (IPLUS START OFFST -1])
                      OLDPTR])
```

(GNC

```
  [LAMBDA (X)                                                (* jop%: "23-Sep-86 16:26")
    (LET ((CODE (GNCCODE X)))
         (AND CODE (FCHARACTER CODE])
```

(GNCCODE

```
  [LAMBDA (X)                                                (* jop%: "23-Sep-86 16:27")
    (COND
       [(STRINGP X)
        (LET ((LEN (fetch (STRINGP LENGTH) of X))
              (OFFST (fetch (STRINGP OFFST) of X)))
             (COND
```

```
               ((NOT (EQ 0 LEN))
                (PROG1 (\GETBASECHAR (ffetch (STRINGP FATSTRINGP) of X)
                             (ffetch (STRINGP BASE) of X)
                             OFFST)
                    (UNINTERRUPTABLY
                        (freplace (STRINGP OFFST) of X with (ADD1 OFFST))
                        (freplace (STRINGP LENGTH) of X with (SUB1 LEN))))]
      (T (NTHCHARCODE X 1])
```

(GLC

```
  [LAMBDA (X)                                                    (* jop%: "23-Sep-86 16:25")
    (LET ((CODE (GLCCODE X)))
         (AND CODE (FCHARACTER CODE])
```

(GLCCODE

```
  [LAMBDA (X)                                                    (* jop%: "23-Sep-86 16:26")
    (COND
       [(STRINGP X)
        (LET [(LEN (SUB1 (fetch (ARRAY-HEADER FILL-POINTER) of X]
             (COND
                ((NOT (EQ -1 LEN))
                 (PROG1 (\GETBASECHAR (ffetch (STRINGP FATSTRINGP) of X)
                              (ffetch (STRINGP BASE) of X)
                              (IPLUS LEN (ffetch (STRINGP OFFST) of X)))
                    (UNINTERRUPTABLY
                        (freplace (ARRAY-HEADER FILL-POINTER-P) of X with T)
                        (freplace (ARRAY-HEADER FILL-POINTER) of X with LEN)))]
      (T (NTHCHARCODE X -1])
```

(STREQUAL

```
  [LAMBDA (X Y)                                                  ; Edited 12-Jan-94 10:07 by sybalsky:mv:envos
    (DECLARE (LOCALVARS . T))
    (AND (STRINGP X)
         (STRINGP Y)
         (PROG ((LEN (ffetch (STRINGP LENGTH) of X)))
               (COND
                  ((NEQ LEN (ffetch (STRINGP LENGTH) of Y))
                   (RETURN)))
               (RETURN (PROG ((BASEX (ffetch (STRINGP BASE) of X))
                              (BNX (ffetch (STRINGP OFFST) of X))
                              (FATPX (ffetch (STRINGP FATSTRINGP) of X))
                              (BASEY (ffetch (STRINGP BASE) of Y))
                              (BNY (ffetch (STRINGP OFFST) of Y))
                              (FATPY (ffetch (STRINGP FATSTRINGP) of Y)))
                             (COND
                                ((OR (NEQ 0 BNX)
                                     (NEQ 0 BNY)
                                     FATPX FATPY)
                                 (GO SLOWLP)))
                         LP  (COND
                                ((EQ 0 LEN)
                                 (RETURN T)))
                             (add LEN -1)
                             (COND
                                ((NEQ (\GETBASEBYTE BASEX LEN)
                                      (\GETBASEBYTE BASEY LEN))
                                 (RETURN)))
                             (GO LP)
                         SLOWLP
                             (COND
                                ((EQ 0 LEN)
                                 (RETURN T))
                                ((NEQ (\GETBASECHAR FATPX BASEX BNX)
                                      (\GETBASECHAR FATPY BASEY BNY))
                                 (RETURN))
                                (T (add BNX 1)
                                   (add BNY 1)
                                   (add LEN -1)
                                   (GO SLOWLP])
```

(STRING.EQUAL

```
  [LAMBDA (X Y CASEARRAY)                                        ; Edited  8-Jan-2022 19:08 by rmk
                                                                 ; Edited 12-Jan-94 10:01 by sybalsky:mv:envos
```

;;; True if X and Y are equal atoms or strings without respect to alphabetic case.

;; RMK: Added CASEARRAY argument, silly not to extend this to other than the default UPPERCASEARRAY.

```
    (PROG (CABASE LEN BASEX OFFSETX FATPX BASEY OFFSETY FATPY C1 C2)
          (COND
             ((LITATOM X)
              (SETQ LEN (ffetch (LITATOM PNAMELENGTH) of X))
              (SETQ BASEX (ffetch (LITATOM PNAMEBASE) of X))
              (SETQ OFFSETX 1)
```

```
             (SETQ FATPX (ffetch (LITATOM FATPNAMEP) of X)))
            ((STRINGP X)
             (SETQ LEN (ffetch (STRINGP LENGTH) of X))
             (SETQ BASEX (ffetch (STRINGP BASE) of X))
             (SETQ OFFSETX (ffetch (STRINGP OFFST) of X))
             (SETQ FATPX (ffetch (STRINGP FATSTRINGP) of X)))
            ((SETQ X (MKSTRING X))
             (SETQ LEN (ffetch (STRINGP LENGTH) of X))
             (SETQ BASEX (ffetch (STRINGP BASE) of X))
             (SETQ OFFSETX (ffetch (STRINGP OFFST) of X))
             (SETQ FATPX (ffetch (STRINGP FATSTRINGP) of X)))
            (T (RETURN NIL)))
         (COND
            ((LITATOM Y)
             (COND
                ((NEQ LEN (ffetch (LITATOM PNAMELENGTH) of Y))
                 (RETURN)))
             (SETQ BASEY (ffetch (LITATOM PNAMEBASE) of Y))
             (SETQ OFFSETY 1)
             (SETQ FATPY (ffetch (LITATOM FATPNAMEP) of Y)))
            ((STRINGP Y)
             (COND
                ((NEQ LEN (ffetch (STRINGP LENGTH) of Y))
                 (RETURN)))
             (SETQ BASEY (ffetch (STRINGP BASE) of Y))
             (SETQ OFFSETY (ffetch (STRINGP OFFST) of Y))
             (SETQ FATPY (ffetch (STRINGP FATSTRINGP) of Y)))
            ((SETQ Y (MKSTRING Y))
             (COND
                ((NEQ LEN (ffetch (STRINGP LENGTH) of Y))
                 (RETURN)))
             (SETQ BASEY (ffetch (STRINGP BASE) of Y))
             (SETQ OFFSETY (ffetch (STRINGP OFFST) of Y))
             (SETQ FATPY (ffetch (STRINGP FATSTRINGP) of Y)))
            (T (RETURN NIL)))
         (CL:UNLESS CASEARRAY (SETQ CASEARRAY UPPERCASEARRAY))
         [COND
            ((NEQ (ffetch (ARRAYP TYP) of (\DTEST CASEARRAY 'ARRAYP))
                  \ST.BYTE)
             (IF (EQ CASEARRAY UPPERCASEARRAY)
                 THEN ;; Did someone smashed the UPPERCASEARRAY? We can repair it
                      (SETQ CASEARRAY (SETQ UPPERCASEARRAY (UPPERCASEARRAY)))
               ELSE (\ILLEGAL.ARG CASEARRAY]
         (SETQ CABASE (ffetch (ARRAYP BASE) of CASEARRAY))
         (RETURN (COND
                    [(OR FATPX FATPY)                          ; Slow case
                     (for BNX from OFFSETX as BNY from OFFSETY as I to LEN
                        always (PROGN (SETQ C1 (\GETBASECHAR FATPX BASEX BNX))
                                      (SETQ C2 (\GETBASECHAR FATPY BASEY BNY))
                                      (COND
                                         ((OR (IGREATERP C1 \MAXTHINCHAR)
                                              (IGREATERP C2 \MAXTHINCHAR))
                                                                ; Fat chars not alphabetic
                                          (EQ C1 C2))
                                         (T (EQ (\GETBASEBYTE CABASE C1)
                                                (\GETBASEBYTE CABASE C2]
                    (T (for BNX from OFFSETX as BNY from OFFSETY as I to LEN
                          always (EQ (\GETBASEBYTE CABASE (\GETBASETHIN BASEX BNX))
                                     (\GETBASEBYTE CABASE (\GETBASETHIN BASEY BNY])
```

(STRINGP

```
[LAMBDA (OBJECT)                                               (* jop%: "24-Sep-86 22:58")
  (AND (%%STRINGP OBJECT)
       OBJECT])
```

(CHCON1

```
[LAMBDA (X)                                                    (* jop%: "23-Sep-86 17:45")
```

;;; This is opencoded NTHCHARCODE* for the case where N=1 and FLG=NIL

```
  (COND
     [(STRINGP X)
      (AND (NEQ (fetch (STRINGP LENGTH) of X)
                0)
           (\GETBASECHAR (fetch (STRINGP FATSTRINGP) of X)
                  (fetch (STRINGP BASE) of X)
                  (fetch (STRINGP OFFST) of X]
     ((LITATOM X)
      (AND (NEQ (ffetch (LITATOM PNAMELENGTH) of X)
                0)
           (\GETBASECHAR (ffetch (LITATOM FATPNAMEP) of X)
                  (ffetch (LITATOM PNAMEBASE) of X)
                  1)))
     (T (NTHCHARCODE X 1])
```

(U-CASE

```
[LAMBDA (X)                                                    ; Edited 11-Nov-2018 13:06 by rmk:
                                                               ; Edited 10-Feb-87 19:12 by bvm:

   (COND
      [(LITATOM X)
       (WITH-RESOURCE (\PNAMESTRING)
              ;; RMK:  This was set up to call \MKATOM with the characters in a fat-string array, even if the original atom was a thin atom. Then
              ;; \MKATOM is suppose to sort it out.  But case-changing in the ASCII range doesn't coerce between fat and thin.  So this should
              ;; use the format of the original atom, not rely on \MKATOM to correct.
              (for C CHANGEFLG (FATP _ (FETCH (LITATOM FATPNAMEP) OF X))
                   (BASE _ (ffetch (STRINGP BASE) of \PNAMESTRING))
                   inatom X as I from 0 do (\PUTBASECHAR FATP BASE I (COND
                                                                         [(AND (IGEQ C (CHARCODE a))
                                                                               (ILEQ C (CHARCODE z)))
                                                                          (SETQ CHANGEFLG
                                                                           (IPLUS C (IDIFFERENCE (CHARCODE A)
                                                                                                 (CHARCODE a]
                                                                         (T C)))
                finally (RETURN (COND
                                   ((OR CHANGEFLG (NEQ (CL:SYMBOL-PACKAGE X)
                                                       *INTERLISP-PACKAGE*))
                                    (\MKATOM BASE 0 I FATP))
                                   (T                                  ; Don't bother calling \MKATOM if X already uppercase and
                                                                       ; interned in IL
                                      X]
      ((STRINGP X)
       (for C BASE NEWSTRING (FATP _ (ffetch (STRINGP FATSTRINGP) of X))
            instring X as I from 0 first (SETQ NEWSTRING (ALLOCSTRING (\NSTRINGCHARS X)
                                                                      NIL NIL FATP))
                                         (SETQ BASE (ffetch (STRINGP XBASE) of NEWSTRING))
          do (\PUTBASECHAR FATP BASE I (COND
                                          [(AND (IGEQ C (CHARCODE a))
                                                (ILEQ C (CHARCODE z)))
                                           (IPLUS C (IDIFFERENCE (CHARCODE A)
                                                                 (CHARCODE a]
                                          (T C)))
          finally (RETURN NEWSTRING)))
      [(LISTP X)
       (CONS (U-CASE (CAR X))
             (AND (CDR X)
                  (U-CASE (CDR X]
      (T X])
```

(L-CASE

```
[LAMBDA (X FLG)                                                ; Edited 11-Nov-2018 13:07 by rmk:
                                                               ; Edited 10-Feb-87 19:12 by bvm:

   ;; RMK:  See comment in U-CASE
   (COND
      [(LITATOM X)
       (WITH-RESOURCE (\PNAMESTRING)
              (for C CHANGEFLG (FATP _ (FETCH (LITATOM FATPNAMEP) OF X))
                   (BASE _ (ffetch (STRINGP XBASE) of \PNAMESTRING))
                   inatom X as I from 0 do [COND
                                              [(AND (IGEQ C (CHARCODE A))
                                                    (ILEQ C (CHARCODE Z)))
                                               (COND
                                                  (FLG (SETQ FLG NIL))
                                                  (T (SETQ CHANGEFLG (SETQ C (IPLUS C (IDIFFERENCE (CHARCODE
                                                                                                       a)
                                                                                                   (CHARCODE A]
                                              ([AND FLG (AND (IGEQ C (CHARCODE a))
                                                             (ILEQ C (CHARCODE z]
                                               (SETQ FLG NIL)
                                               (SETQ CHANGEFLG (SETQ C (IPLUS C (IDIFFERENCE (CHARCODE A)
                                                                                             (CHARCODE a]
                                              (\PUTBASECHAR FATP BASE I C)
                finally (RETURN (COND
                                   ((OR CHANGEFLG (NEQ (CL:SYMBOL-PACKAGE X)
                                                       *INTERLISP-PACKAGE*))
                                    (\MKATOM BASE 0 I FATP))
                                   (T                                  ; Don't bother calling \MKATOM if X already lowercase and
                                                                       ; interned in IL
                                      X]
      ((STRINGP X)
       (for C BASE NEWSTRING (FATP _ (ffetch (STRINGP FATSTRINGP) of X))
            instring X as I from 0 first (SETQ NEWSTRING (ALLOCSTRING (\NSTRINGCHARS X)
                                                                      NIL NIL FATP))
                                         (SETQ BASE (ffetch (STRINGP BASE) of NEWSTRING))
          do [COND
                [(AND (IGEQ C (CHARCODE A))
                      (ILEQ C (CHARCODE Z)))
                 (COND
```

```
                    (FLG (SETQ FLG NIL))
                    (T (SETQ C (IPLUS C (IDIFFERENCE (CHARCODE a)
                                                     (CHARCODE A]
                ([AND FLG (AND (IGEQ C (CHARCODE a))
                               (ILEQ C (CHARCODE z]
                 (SETQ FLG NIL)
                 (SETQ C (IPLUS C (IDIFFERENCE (CHARCODE A)
                                               (CHARCODE a]
            (\PUTBASECHAR FATP BASE I C)
        finally (RETURN NEWSTRING)))
      [(LISTP X)
       (CONS (L-CASE (CAR X)
                     FLG)
             (AND (CDR X)
                  (L-CASE (CDR X)
                          FLG]
      (T X])
```

(U-CASEP

```
(U-CASEP
  [LAMBDA (X)                                                    (* jop%: "23-Sep-86 16:43")
    (COND
       [(LITATOM X)
        (for C inatom X never (AND (IGEQ C (CHARCODE a))
                                   (ILEQ C (CHARCODE z]
       [(STRINGP X)
        (for C instring X never (AND (IGEQ C (CHARCODE a))
                                     (ILEQ C (CHARCODE z]
       [(LISTP X)
        (AND (U-CASEP (CAR X))
             (OR (NULL (CDR X))
                 (U-CASEP (CDR X]
       (T T])
```

(\SMASHABLESTRING

```
(\SMASHABLESTRING
  [LAMBDA (STR FATP)                                             (* gbn "18-Apr-85 00:39")
    ;; Ensures that FATP characters can be smashed into STR
    (COND
       [(ffetch (STRINGP READONLY) of STR)
        (\MAKEWRITABLESTRING STR (OR FATP (ffetch (STRINGP FATSTRINGP) of STR]
       ((AND FATP (NOT (ffetch (STRINGP FATSTRINGP) of STR)))
        (\FATTENSTRING STR)))
    STR])
```

(\MAKEWRITABLESTRING

```
(\MAKEWRITABLESTRING
  [LAMBDA (STR NEWFATP)                                          (* jop%: "23-Sep-86 16:44")

;;; takes a string pointing at a readonly pname and changes the string to point to a block of writable memory of the appropriate width

    (%%MAKE-ARRAY-WRITEABLE STR)
    (if (AND NEWFATP (NOT (ffetch (STRINGP FATSTRINGP) of STR)))
        then (%%MAKE-STRING-ARRAY-FAT STR))
    STR])
```

(\SMASHSTRING

```
(\SMASHSTRING
  [LAMBDA (DEST POS SOURCE NC)                                   (* jop%: "23-Sep-86 16:51")

;;; copy NC characters from the string SOURCE to the string DEST starting at character POS (counting from 0) of DEST.  If NC=NIL, length of SOURCE
;;; is used.  DEST is presumed to be not READONLY, long enough for the smash, and to be fat if SOURCE contains any fat characters--the caller must
;;; guarantee this.
                                                                 ; Only caller so far is \RSTRING2 in the reader
    (OR NC (SETQ NC (ffetch (STRINGP LENGTH) of SOURCE)))
    (add POS (ffetch (STRINGP OFFST) of DEST))
    (COND
       [(ffetch (STRINGP FATSTRINGP) of DEST)                    ; The destination is fat.
        (COND
           ((ffetch (STRINGP FATSTRINGP) of SOURCE)              ; The source is also;  just copy the characters straight across
            (\BLT (\ADDBASE (ffetch (STRINGP BASE) of DEST)
                            POS)
                  (\ADDBASE (ffetch (STRINGP BASE) of SOURCE)
                            (ffetch (STRINGP OFFST) of SOURCE))
                  NC))
           (T                                                    ; Have to do thin-to-fat conversion
              (bind (DBASE _ (ffetch (STRINGP BASE) of DEST)) for C inthinstring SOURCE as DESTCH# from POS
                 as SRCH# from 1 to NC do                        ; Run thru chars 1..NC (or len) of the source, moving them into
                                                                 ; the destination
                                         (\PUTBASEFAT DBASE DESTCH# C]
       ((ffetch (STRINGP FATSTRINGP) of SOURCE)                  ; Assume that SOURCE is FATP with no fat characters.  This is
                                                                 ; a guarantee made by \RSTRING2.
        (bind (DBASE _ (ffetch (STRINGP BASE) of DEST)) for C infatstring SOURCE as DESTCH# from POS as SRCH#
           from 1 to NC do                                       ; Run thru chars 1..NC (or len) of the source, moving them into
                                                                 ; the destination
                         (AND (IGREATERP C \MAXTHINCHAR)
```

```
                              (SHOULDNT))                          ; If we find an unexpected fat character, complain!
                          (\PUTBASETHIN DBASE DESTCH# C)))
       (T                                                          ; The source and destination are both thin.  Just copy characters.
          (\MOVEBYTES (ffetch (STRINGP BASE) of SOURCE)
                 (ffetch (STRINGP OFFST) of SOURCE)
                 (ffetch (STRINGP BASE) of DEST)
                 POS NC)))
    DEST])
```

(\FATTENSTRING

```
  [LAMBDA (STR)                                                    (* jop%: "11-Sep-86 18:00")
    (%%MAKE-STRING-ARRAY-FAT STR])

)

;; Temporary until low level system is changed to call STRING.EQUAL again

(MOVD? 'STRING.EQUAL 'STRING-EQUAL NIL T)

(MOVD? 'STRING.EQUAL 'CL::SIMPLE-STRING-EQUAL NIL T)

(DEFINEQ
```

(\GETBASESTRING

```
  [LAMBDA (BASE BYTEOFFSET NCHARS FATP)                            (* jop%: "23-Sep-86 17:50")
```

;;; Makes a string consisting of NCHARS characters starting at BYTEOFFSET from BASE -- note that caller must know whether the string is fat (see ;;; \PUTBASESTRING); BYTEOFFSET is always a byte offset in either case

```
    (LET ((STR (ALLOCSTRING NCHARS NIL NIL FATP)))
         (\MOVEBYTES BASE BYTEOFFSET (fetch (STRINGP BASE) of STR)
                (fetch (STRINGP OFFST) of STR)
                (COND
                   (FATP (UNFOLD NCHARS BYTESPERWORD))
                   (T NCHARS)))
         STR])
```

(\PUTBASESTRING

```
  [LAMBDA (BASE BYTEOFFSET SOURCE FATP)                            (* jop%: "23-Sep-86 16:48")
```

;; In addition to putting the bytes into memory, this guy returns the number of characters 'written' , since the source may not be a STRINGP, but will ;; be coerced to one.

;; Not clear what this fn should do with fat strings. Caller is using this fn to store raw characters into some random location, so must make some ;; assumption about the format they are stored in. Hence if there's a fat string, but FATP is false, we don't know what to do

```
    (COND
       ((STRINGP SOURCE)
        (COND
           (FATP (\PUTBASESTRINGFAT BASE BYTEOFFSET (ffetch (STRINGP BASE) of SOURCE)
                        (ffetch (STRINGP OFFST) of SOURCE)
                        (ffetch (STRINGP LENGTH) of SOURCE)
                        (ffetch (STRINGP FATSTRINGP) of SOURCE)))
           ((ffetch (STRINGP FATSTRINGP) of SOURCE)
            [for CH infatstring SOURCE as OFFSET from BYTEOFFSET do (COND
                                                                      ((ILEQ CH \MAXTHINCHAR)
                                                                       (\PUTBASEBYTE BASE OFFSET CH))
                                                                      (T (ERROR "Fat string in \PUTBASESTRING"
                                                                                SOURCE]
            (ffetch (STRINGP LENGTH) of SOURCE))
           (T (\MOVEBYTES (ffetch (STRINGP BASE) of SOURCE)
                      (ffetch (STRINGP OFFST) of SOURCE)
                      BASE BYTEOFFSET (SETQ SOURCE (ffetch (STRINGP LENGTH) of SOURCE)))
              SOURCE)))
       ((LITATOM SOURCE)
        (COND
           (FATP (\PUTBASESTRINGFAT BASE BYTEOFFSET (ffetch (LITATOM PNAMEBASE) of SOURCE)
                        1
                        (ffetch (LITATOM PNAMELENGTH) of SOURCE)
                        (ffetch (LITATOM FATPNAMEP) of SOURCE)))
           ((ffetch (LITATOM FATPNAMEP) of SOURCE)
            [for CH infatatom SOURCE as OFFSET from BYTEOFFSET do (COND
                                                                    ((ILEQ CH \MAXTHINCHAR)
                                                                     (\PUTBASEBYTE BASE OFFSET CH))
                                                                    (T (ERROR "Fat string in \PUTBASESTRING"
                                                                              SOURCE]
            (ffetch (LITATOM PNAMELENGTH) of SOURCE))
           (T (\MOVEBYTES (ffetch (LITATOM PNAMEBASE) of SOURCE)
                      1 BASE BYTEOFFSET (SETQ SOURCE (ffetch (LITATOM PNAMELENGTH) of SOURCE)))
              SOURCE)))
       (T (\PUTBASESTRING BASE BYTEOFFSET (MKSTRING SOURCE)
                FATP])
```

(\PUTBASESTRINGFAT

```
  [LAMBDA (DBASE DBYTEOFFSET SBASE SOFFSET LEN FATP)               (* jop%: " 8-Sep-86 21:02")
```

;;; Store a fat string at byte offset from DBASE. SBASE and SOFFSET are in the source's units (bytes or words)

```
    [COND
       (FATP (\MOVEBYTES SBASE (UNFOLD SOFFSET BYTESPERWORD)
                     DBASE DBYTEOFFSET (UNFOLD LEN BYTESPERWORD)))
       (T                                                          ; Store thin string in fat format
          (for I from 0 to (SUB1 LEN) as DOFF from DBYTEOFFSET by 2 do (\PUTBASETHIN DBASE DOFF 0)
                                                                       (\PUTBASETHIN DBASE (ADD1 DOFF)
                                                                              (\GETBASETHIN SBASE (IPLUS SOFFSET I
                                                                                                         ]
    LEN])
```

(**GetBcplString**

```
  [LAMBDA (BASE ATOMFLG)                                         (* jop%: "23-Sep-86 17:46")
```

;; Returns as a Lisp string the Bcpl string stored at BS. Format is one byte length, follwed by chars. If ATOMFLG is true, returns result as an atom

```
    (LET ((L (\GETBASEBYTE BASE 0))
          S)
         (COND
            ((AND ATOMFLG (ILEQ L \PNAMELIMIT))
             (\MKATOM BASE 1 L))
            (T (SETQ S (\GETBASESTRING BASE 1 L))
               (COND
                  (ATOMFLG                                       ; Let MKATOM handle the error
                         (MKATOM S))
                  (T S])
```

(**SetBcplString**

```
  [LAMBDA (BASE STR)                                             (* bvm%: " 5-Jul-85 21:50")
    (LET ((L (NCHARS STR)))
         (COND
            ((IGREATERP L 255)
             (LISPERROR "ILLEGAL ARG" BASE))
            (T (\PUTBASEBYTE BASE 0 L)
               (\PUTBASESTRING BASE 1 STR)))
         BASE])

)

(DECLARE%: DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(ACCESSFNS STRINGP ((XREADONLY (fetch (ARRAY-HEADER READ-ONLY-P) of DATUM)
                               (replace (ARRAY-HEADER READ-ONLY-P) of DATUM with NEWVALUE))
                    (XBASE ([OPENLAMBDA (STRING)
                                (COND
                                   ((fetch (ARRAY-HEADER INDIRECT-P) of STRING)
                                    (%%ARRAY-BASE STRING))
                                   (T (fetch (ARRAY-HEADER BASE) of STRING]
                            DATUM)
                           ((OPENLAMBDA (STRING NV)
                              (replace (ARRAY-HEADER INDIRECT-P) of STRING with NIL)
                              (replace (ARRAY-HEADER BASE) of STRING with NV)
                              NV)
                            DATUM NEWVALUE))
                    (TYP ((OPENLAMBDA (STRING)
                            (SELECTC (COND
                                        ((fetch (ARRAY-HEADER INDIRECT-P) of STRING)
                                         (%%ARRAY-TYPE-NUMBER STRING))
                                        (T (fetch (ARRAY-HEADER TYPE-NUMBER) of STRING)))
                                     (%%THIN-CHAR-TYPENUMBER
                                         \ST.BYTE)
                                     (%%FAT-CHAR-TYPENUMBER
                                         \ST.POS16)
                                     (SHOULDNT "Unknown type-number")))
                          DATUM)
                         ([OPENLAMBDA (STRING NV)
                            (LET [(%%NEW-TYPE-NUMBER (SELECTC NV
                                                              (\ST.BYTE %%THIN-CHAR-TYPENUMBER)
                                                              (\ST.POS16 %%FAT-CHAR-TYPENUMBER)
                                                              (SHOULDNT "Unknown typ value"]
                                  (COND
                                     ((fetch (ARRAY-HEADER INDIRECT-P) of STRING)
                                      (%%SET-ARRAY-TYPE-NUMBER STRING %%NEW-TYPE-NUMBER))
                                     (T (replace (ARRAY-HEADER TYPE-NUMBER) of STRING with %%NEW-TYPE-NUMBER]
                          DATUM NEWVALUE))
                    (LENGTH (fetch (ARRAY-HEADER FILL-POINTER) of DATUM)
                            ((OPENLAMBDA (STRING NV)
                               (replace (ARRAY-HEADER FILL-POINTER) of STRING with NV)
                               (replace (ARRAY-HEADER TOTAL-SIZE) of STRING with NV)
                               [COND
```

```
                                ((%%GENERAL-ARRAY-P STRING)
                                 (freplace (GENERAL-ARRAY DIMS) of STRING with (LIST NV]
                               NV)
                              DATUM NEWVALUE))
                     (OFFST ([OPENLAMBDA (STRING)
                                (COND
                                   ((fetch (ARRAY-HEADER INDIRECT-P) of STRING)
                                    (%%ARRAY-OFFSET STRING))
                                   (T (fetch (ARRAY-HEADER OFFSET) of STRING]
                              DATUM)
                             ([OPENLAMBDA (STRING NV)
                                (COND
                                   ((NOT (EQ 0 NV))
                                    (replace (ARRAY-HEADER DISPLACED-P) of STRING with T)))
                                (COND
                                   ((fetch (ARRAY-HEADER INDIRECT-P) of STRING)
                                    (%%SET-ARRAY-OFFSET STRING NV))
                                   (T (replace (ARRAY-HEADER OFFSET) of STRING with NV]
                              DATUM NEWVALUE))
                     ;; The rest of these fields only appear when smashing
                     (XFLAGS (LOGAND (fetch (ARRAY-HEADER FLAGS) of DATUM)
                                     15)
                             ((OPENLAMBDA (STRING)
                                (replace (ARRAY-HEADER ADJUSTABLE-P) of STRING with NIL)
                                (replace (ARRAY-HEADER DISPLACED-P) of STRING with NIL)
                                (replace (ARRAY-HEADER FILL-POINTER-P) of STRING with NIL)
                                (replace (ARRAY-HEADER EXTENDABLE-P) of STRING with NIL))
                              DATUM)))
       [ACCESSFNS STRINGP ((ORIG ((OPENLAMBDA (STRING)
                                    1)
                                  DATUM)
                                 ((OPENLAMBDA (STRING NV)
                                    (COND
                                       ((NOT (EQ NV 1))
                                        (ERROR "Il:stringp's are always origin 1")))
                                    NV)
                                  DATUM NEWVALUE))                ; An inoperative field
                           (SUBSTRINGED ((OPENLAMBDA (STRING)
                                           NIL)
                                         DATUM)
                                  ((OPENLAMBDA (STRING NV)
                                     (OR (NULL NV)
                                         (ERROR "Substringed field not supported")))
                                   DATUM NEWVALUE))
                           (READONLY (ffetch (STRINGP XREADONLY) of DATUM)
                                  (freplace (STRINGP XREADONLY) of DATUM with NEWVALUE))
                           (FATSTRINGP ((OPENLAMBDA (STRING)
                                          (EQ (COND
                                                 ((fetch (ARRAY-HEADER INDIRECT-P) of STRING)
                                                  (%%ARRAY-TYPE-NUMBER STRING))
                                                 (T (fetch (ARRAY-HEADER TYPE-NUMBER) of STRING)))
                                              %%FAT-CHAR-TYPENUMBER))
                                        DATUM)
                                  ([OPENLAMBDA (STRING NV)
                                     (LET [(%%NEW-TYPE-NUMBER (COND
                                                                 (NV %%FAT-CHAR-TYPENUMBER)
                                                                 (T %%THIN-CHAR-TYPENUMBER]
                                       (COND
                                          ((fetch (ARRAY-HEADER INDIRECT-P) of STRING)
                                           (%%SET-ARRAY-TYPE-NUMBER STRING %%NEW-TYPE-NUMBER))
                                          (T (replace (ARRAY-HEADER TYPE-NUMBER) of STRING with
                                                                                    %%NEW-TYPE-NUMBER
                                                                                    ]
                                   DATUM NEWVALUE))
                           (BASE (ffetch (STRINGP XBASE) of DATUM)
                                 (freplace (STRINGP XBASE) of DATUM with NEWVALUE]
       (CREATE (create ONED-ARRAY
                       BASE _ XBASE
                       READ-ONLY-P _ XREADONLY
                       STRING-P _ T
                       DISPLACED-P _ (NOT (EQ OFFST 0))
                       TYPE-NUMBER _ (COND
                                        ((EQ TYP \ST.POS16)
                                         %%FAT-CHAR-TYPENUMBER)
                                        (T %%THIN-CHAR-TYPENUMBER))
                       OFFSET _ OFFST
                       FILL-POINTER _ LENGTH
                       TOTAL-SIZE _ LENGTH))
       (TYPE? (CL:STRINGP DATUM))
       OFFST _ 0 TYP _ \ST.BYTE LENGTH _ 0)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \OneCharAtomBase)
)
```

```
(DECLARE%: EVAL@COMPILE

[PUTDEF '\NUMSTR 'RESOURCES '(NEW (ALLOCSTRING 128]

[PUTDEF '\NUMSTR1 'RESOURCES '(NEW (CONCAT]

[PUTDEF '\PNAMESTRING 'RESOURCES '(NEW (ALLOCSTRING \PNAMELIMIT NIL NIL \FATPNAMESTRINGP]
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \FATPNAMESTRINGP T)

(CONSTANTS (\FATPNAMESTRINGP T))
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \PNAMESTRINGPUTCHAR MACRO ((BASE OFFSET CODE)       ; For stuffing chars into resource \PNAMESTRING
                                     (\PUTBASECHAR \FATPNAMESTRINGP BASE OFFSET CODE)))
)


(DEFOPTIMIZER FCHARACTER (NUM)
                         `((LAMBDA (N)
                             (COND
                                ((IGREATERP N \MAXTHINCHAR) ; The character we're getting is NOT a thin character -- do it the
                                                            ; hard way
                                 (CHARACTER N))
                                ((IGREATERP N (CHARCODE 9))
                                 (\ADDBASE \OneCharAtomBase (IDIFFERENCE N 10)))
                                ((IGEQ N (CHARCODE 0))
                                 (IDIFFERENCE N (CHARCODE 0)))
                                (T                          ; The common case -- just add on the one-atom base.
                                   (\ADDBASE \OneCharAtomBase N]
                           ,NUM))

(DECLARE%: EVAL@COMPILE

(I.S.OPR 'inpname NIL '[SUBST (GETDUMMYVAR)
                         '$$BODY
                         `(bind $$OFFSET _ 1 $$BODY _ BODY $$BASE $$END $$FATP $$READONLY
                             declare (LOCALVARS $$END $$BODY $$FATP $$BASE $$OFFSET $$READONLY)
                             first [PROG NIL
                                    $$RETRY
                                         (COND
                                            ((STRINGP $$BODY)
                                             (SETQ $$BASE (ffetch (STRINGP BASE) of $$BODY))
                                             (SETQ $$OFFSET (ffetch (STRINGP OFFST) of $$BODY))
                                             (SETQ $$END (IPLUS $$OFFSET -1 (ffetch (STRINGP LENGTH)
                                                                               of $$BODY)))
                                             (SETQ $$FATP (ffetch (STRINGP FATSTRINGP) of $$BODY))
                                             (SETQ $$READONLY (ffetch (STRINGP READONLY) OF $$BODY)))
                                            ((LITATOM $$BODY)
                                             (SETQ $$BASE (ffetch (LITATOM PNAMEBASE) of $$BODY))
                                             (SETQ $$END (ffetch (PNAMEBASE PNAMELENGTH) of $$BASE))
                                             (SETQ $$FATP (ffetch (LITATOM FATPNAMEP) of $$BODY))
                                             (SETQ $$READONLY T))
                                            (T (SETQ $$BODY (MKSTRING $$BODY))
                                               (GO $$RETRY]
                             eachtime (AND (IGREATERP $$OFFSET $$END)
                                           (GO $$OUT))
                                      (SETQ I.V. (\GETBASECHAR $$FATP $$BASE $$OFFSET))
                             repeatwhile (PROGN (SETQ $$OFFSET (ADD1 $$OFFSET))
                                                T]
       T)

(I.S.OPR 'inatom NIL '[SUBST (GETDUMMYVAR)
                         '$$BODY
                         '(bind $$OFFSET _ 1 $$BODY _ BODY $$BASE $$END $$FATP
                             declare (LOCALVARS $$OFFSET $$BODY $$BASE $$END $$FATP)
                             first (SETQ $$BASE (ffetch (LITATOM PNAMEBASE) of $$BODY))
                                  (SETQ $$END (ffetch (PNAMEBASE PNAMELENGTH) of $$BASE))
                                  (SETQ $$FATP (ffetch (LITATOM FATPNAMEP) of $$BODY))
                             eachtime (AND (IGREATERP $$OFFSET $$END)
                                           (GO $$OUT))
                                      (SETQ I.V. (\GETBASECHAR $$FATP $$BASE $$OFFSET))
                             repeatwhile (PROGN (SETQ $$OFFSET (ADD1 $$OFFSET))
                                                T]
       T)

(I.S.OPR 'instring NIL '[SUBST (GETDUMMYVAR)
                         '$$BODY
                         '(bind $$BODY _ BODY $$END $$OFFSET $$BASE $$FATP
                             declare (LOCALVARS $$BODY $$END $$OFFSET $$BASE $$FATP)
                             first (SETQ $$OFFSET (ffetch (STRINGP OFFST) of $$BODY))
```

```
                                        (SETQ $$BASE (ffetch (STRINGP BASE) of $$BODY))
                                        (SETQ $$END (IPLUS $$OFFSET -1 (ffetch (STRINGP LENGTH) of $$BODY)))
                                        (SETQ $$FATP (ffetch (STRINGP FATSTRINGP) of $$BODY))
                                    eachtime (AND (IGREATERP $$OFFSET $$END)
                                                  (GO $$OUT))
                                      (SETQ I.V. (\GETBASECHAR $$FATP $$BASE $$OFFSET))
                                    repeatwhile (PROGN (SETQ $$OFFSET (ADD1 $$OFFSET))
                                                       T]
       T)
)

(DECLARE%: EVAL@COMPILE

(I.S.OPR 'infatatom NIL '[SUBST (GETDUMMYVAR)
                               '$$BODY
                               '(bind $$OFFSET _ 1 $$BODY _ BODY $$BASE $$END
                                   declare (LOCALVARS $$OFFSET $$BODY $$BASE $$END)
                                   first (SETQ $$BASE (ffetch (LITATOM PNAMEBASE) of $$BODY))
                                        (SETQ $$END (ffetch (PNAMEBASE PNAMELENGTH) of $$BASE))
                                   eachtime (AND (IGREATERP $$OFFSET $$END)
                                                 (GO $$OUT))
                                     (SETQ I.V. (\GETBASEFAT $$BASE $$OFFSET))
                                   repeatwhile (PROGN (SETQ $$OFFSET (ADD1 $$OFFSET))
                                                      T]
       T)

(I.S.OPR 'inthinatom NIL '[SUBST (GETDUMMYVAR)
                                '$$BODY
                                '(bind $$OFFSET _ 1 $$BODY _ BODY $$BASE $$END
                                    declare (LOCALVARS $$OFFSET $$BODY $$BASE $$END)
                                    first (SETQ $$BASE (ffetch (LITATOM PNAMEBASE) of $$BODY))
                                         (SETQ $$END (ffetch (PNAMEBASE PNAMELENGTH) of $$BASE))
                                    eachtime (AND (IGREATERP $$OFFSET $$END)
                                                  (GO $$OUT))
                                      (SETQ I.V. (\GETBASETHIN $$BASE $$OFFSET))
                                    repeatwhile (PROGN (SETQ $$OFFSET (ADD1 $$OFFSET))
                                                       T]
       T)

(I.S.OPR 'infatstring NIL '[SUBST (GETDUMMYVAR)
                                 '$$BODY
                                 '(bind $$BODY _ BODY $$END $$OFFSET $$BASE
                                     declare (LOCALVARS $$BODY $$END $$OFFSET $$BASE)
                                     first (SETQ $$OFFSET (ffetch (STRINGP OFFST) of $$BODY))
                                          (SETQ $$BASE (ffetch (STRINGP BASE) of $$BODY))
                                          (SETQ $$END (IPLUS $$OFFSET -1 (ffetch (STRINGP LENGTH) of $$BODY)))
                                     eachtime (AND (IGREATERP $$OFFSET $$END)
                                                   (GO $$OUT))
                                       (SETQ I.V. (\GETBASEFAT $$BASE $$OFFSET))
                                     repeatwhile (PROGN (SETQ $$OFFSET (ADD1 $$OFFSET))
                                                        T]
       T)

(I.S.OPR 'inthinstring NIL '[SUBST (GETDUMMYVAR)
                                  '$$BODY
                                  '(bind $$BODY _ BODY $$END $$OFFSET $$BASE
                                      declare (LOCALVARS $$BODY $$END $$OFFSET $$BASE)
                                      first (SETQ $$OFFSET (ffetch (STRINGP OFFST) of $$BODY))
                                           (SETQ $$BASE (ffetch (STRINGP BASE) of $$BODY))
                                           (SETQ $$END (IPLUS $$OFFSET -1 (ffetch (STRINGP LENGTH) of $$BODY)))
                                      eachtime (AND (IGREATERP $$OFFSET $$END)
                                                    (GO $$OUT))
                                        (SETQ I.V. (\GETBASETHIN $$BASE $$OFFSET))
                                      repeatwhile (PROGN (SETQ $$OFFSET (ADD1 $$OFFSET))
                                                         T]
       T)
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \CHARCODEP DMACRO (OPENLAMBDA (X)                    ; used to also say (ILEQ X \MAXFATCHAR), but that's implied by
                                                               ; the first two clauses
                              (AND (SMALLP X)
                                   (IGEQ X 0))))

(PUTPROPS \FATCHARCODEP DMACRO (OPENLAMBDA (X)                 ; Used to also say (ILEQ X \MAXFATCHAR), but that's implied
                                                               ; by the first two clauses
                                 (AND (SMALLP X)
                                      (IGREATERP X \MAXTHINCHAR))))

(PUTPROPS \THINCHARCODEP DMACRO (OPENLAMBDA (X)
                                  (AND (SMALLP X)
                                       (IGEQ X 0)
                                       (ILEQ X \MAXTHINCHAR))))
)

(DECLARE%: EVAL@COMPILE
```

```
(PUTPROPS \GETBASEFAT MACRO (= . \GETBASE))

(PUTPROPS \GETBASETHIN MACRO (= . \GETBASEBYTE))

(PUTPROPS \PUTBASEFAT MACRO (= . \PUTBASE))

(PUTPROPS \PUTBASETHIN MACRO (= . \PUTBASEBYTE))
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \PUTBASECHAR MACRO [OPENLAMBDA (FATP BASE OFFSET CODE)
                                (COND
                                   (FATP (\PUTBASEFAT BASE OFFSET CODE))
                                   (T (\PUTBASETHIN BASE OFFSET CODE])

(PUTPROPS \GETBASECHAR MACRO [(FATP BASE N)
                              (COND
                                 (FATP (\GETBASEFAT BASE N))
                                 (T (\GETBASETHIN BASE N])
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \CHARSET MACRO ((CHARCODE)
                          (LRSH CHARCODE 8)))

(PUTPROPS \CHAR8CODE MACRO ((CHARCODE)
                            (LOGAND CHARCODE 255)))
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \CHARMASK 255)

(RPAQQ \MAXCHAR 255)

(RPAQQ \MAXTHINCHAR 255)

(RPAQQ \MAXFATCHAR 65535)

(RPAQQ \MAXCHARSET 255)

(RPAQQ %#STRINGPWORDS 4)

(CONSTANTS (\CHARMASK 255)
       (\MAXCHAR 255)
       (\MAXTHINCHAR 255)
       (\MAXFATCHAR 65535)
       (\MAXCHARSET 255)
       (%#STRINGPWORDS 4))
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \NATOMCHARS DMACRO ((AT)
                              (fetch (LITATOM PNAMELENGTH) of AT)))

(PUTPROPS \NSTRINGCHARS DMACRO ((S)
                                (fetch (STRINGP LENGTH) of S)))
)
)

;; END EXPORTED DEFINITIONS

(/SETTOPVAL '\\NUMSTR.GLOBALRESOURCE NIL)

(/SETTOPVAL '\\NUMSTR1.GLOBALRESOURCE NIL)

(/SETTOPVAL '\\PNAMESTRING.GLOBALRESOURCE NIL)

(MOVD? 'CHARACTER 'FCHARACTER NIL T)

(DEFINEQ

(%%COPY-ONED-ARRAY
  [LAMBDA (LOCAL-ARRAY)                                    (* jop%: "24-Sep-86 17:51")
    (PROG ((SIZE (LOCAL (ffetch (ONED-ARRAY TOTAL-SIZE) of LOCAL-ARRAY)))
           (BASE (LOCAL (ffetch (ONED-ARRAY BASE) of LOCAL-ARRAY)))
           (OFFSET (LOCAL (ffetch (ONED-ARRAY OFFSET) of LOCAL-ARRAY)))
           (TYPENUMBER (LOCAL (ffetch (ONED-ARRAY TYPE-NUMBER) of LOCAL-ARRAY)))
           NCELLS REMOTE-ARRAY REMOTE-BASE)
          (if (NEQ OFFSET 0)
              then (ERROR "Can't copy an array with non-zero offset"))
          (if (EQ (%%TYPENUMBER-TO-GC-TYPE TYPENUMBER)
                  PTRBLOCK.GCT)
```

```
            then (ERROR "Can't copy pointer arrays"))
          (SETQ NCELLS (FOLDHI (ITIMES (IPLUS SIZE OFFSET)
                                       (%%TYPENUMBER-TO-BITS-PER-ELEMENT TYPENUMBER))
                               BITSPERCELL))
          (SETQ REMOTE-ARRAY (create ONED-ARRAY
                                     BASE _ (\ALLOCBLOCK NCELLS)
                                     STRING-P _ (%%CHAR-TYPE-P TYPENUMBER)
                                     FILL-POINTER-P _ (LOCAL (ffetch (ONED-ARRAY FILL-POINTER-P) of LOCAL-ARRAY))
                                     TYPE-NUMBER _ TYPENUMBER
                                     FILL-POINTER _ (LOCAL (ffetch (ONED-ARRAY FILL-POINTER) of LOCAL-ARRAY))
                                     TOTAL-SIZE _ SIZE))
          (SETQ REMOTE-BASE (ffetch (ONED-ARRAY BASE) of REMOTE-ARRAY))
          [for I from 0 to (SUB1 (LLSH NCELLS 1)) do (\PUTBASE REMOTE-BASE I (LOCAL (\GETBASE BASE I]
          (RETURN REMOTE-ARRAY])
```

(%%COPY-STRING-TO-ARRAY

```
  [LAMBDA (LOCAL-STRING)                                    (* jop%: "24-Sep-86 17:51")
```

;;; Only handles thin strings

```
    (PROG ((SIZE (LOCAL (NCHARS LOCAL-STRING)))
           REMOTE-BASE REMOTE-ARRAY)
          (SETQ REMOTE-BASE (\ALLOCBLOCK (FOLDHI (ITIMES SIZE 8)
                                                 BITSPERCELL)))
          (SETQ REMOTE-ARRAY
           (create ONED-ARRAY
                   BASE _ REMOTE-BASE
                   STRING-P _ T
                   TYPE-NUMBER _ %%THIN-CHAR-TYPENUMBER
                   FILL-POINTER _ SIZE
                   TOTAL-SIZE _ SIZE))
          [for I from 0 to (SUB1 SIZE) do (\PUTBASEBYTE REMOTE-BASE I (LOCAL (NTHCHARCODE LOCAL-STRING
                                                                                  (ADD1 I]
          (RETURN REMOTE-ARRAY])

)

;; For MAKEINIT

(DECLARE%: DONTCOPY

(ADDTOVAR INEWCOMS (FNS ALLOCSTRING %%COPY-ONED-ARRAY %%COPY-STRING-TO-ARRAY))

(ADDTOVAR INEWCOMS (FILES (SYSLOAD FROM VALUEOF DIRECTORIES)
                          CMLARRAY-SUPPORT))

(ADDTOVAR EXPANDMACROFNS \PUTBASETHIN \PUTBASEFAT \CHARCODEP \GETBASECHAR \GETBASETHIN \GETBASEFAT
                         \PUTBASECHAR)

(ADDTOVAR DONTCOMPILEFNS %%COPY-ONED-ARRAY %%COPY-STRING-TO-ARRAY)
)

(DECLARE%: DONTCOPY EVAL@COMPILE

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(LOCALVARS . T)
)
)
```

;; Arrange for the proper compiler

```
(PUTPROPS LLCHAR FILETYPE :FAKE-COMPILE-FILE)

(PUTPROPS LLCHAR COPYRIGHT ("Venue & Xerox Corporation" 1982 1983 1984 1985 1986 1987 1988 1990 1994 2018 2021
                           ))
```

FUNCTION INDEX

MACRO INDEX

CONSTANT INDEX

I.S.OPR INDEX

VARIABLE INDEX

PROPERTY INDEX

RECORD INDEX

OPTIMIZER INDEX