File created: 7-Oct-87 18:15:20 **{POGO:AISNORTH:XEROX}<CUTTING>LISP>UUENCODE.;4**

changes to: (VARS UUENCODECOMS)
(FUNCTIONS UU.SIXBITS UU.LSH)
(FNS UUDECODE-BEGIN-LINE-INTERNAL UUENCODE-INTERNAL UUDECODE-INTERNAL UUDECODE-BODY UUENCODE
UUENCODE-ONE-FILE UUDECODE UUDECODE-BEGIN-LINE UU.TEDIT-INCLUDE-ENCODED UU.DECODE-FROM-TEDIT)

previous date: 7-Oct-87 17:46:11 {POGO:AISNORTH:XEROX}<CUTTING>LISP>UUENCODE.;3

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS


```
(RPAQQ UUENCODECOMS
       (;; UNIX compatible uuencode & uudecode.
        [COMS ;; encoding
              (FNS UUENCODE UUENCODE-ONE-FILE UUENCODE-INTERNAL)
              (INITVARS (UU.MODE-DEFAULT 420))
              (GLOBALVARS UU.MODE-DEFAULT)
              (FUNCTIONS UU.BYTE)
              (DECLARE%: EVAL@COMPILE DONTCOPY (CONSTANTS (UU.CHARS-PER-LINE 45)
                                                          (UU.LAST-TWO-BITS 3)
                                                          (UU.LAST-FOUR-BITS 15)
                                                          (UU.FIRST-TWO-BITS 192]
        (COMS ;; decoding
              (FNS UUDECODE UUDECODE-INTERNAL UUDECODE-BEGIN-LINE UUDECODE-BEGIN-LINE-INTERNAL UUDECODE-BODY)
              (FUNCTIONS UU.SIXBITS UU.LSH)
              [VARS (UU.READTABLE (COPYREADTABLE 'ORIG]
              (GLOBALVARS UU.READTABLE)
              (P (SETBRK NIL NIL UU.READTABLE)
                 (SETSEPR (CHARCODE (SPACE CR LF))
                        NIL UU.READTABLE)
                 (ESCAPE NIL UU.READTABLE)))
        (COMS ;; TEdit interface
              (FNS UU.TEDIT-INCLUDE-ENCODED UU.DECODE-FROM-TEDIT)
              [P (AND (FGETD 'TEDIT)
                      (TEDIT.ADD.MENUITEM TEDIT.DEFAULT.MENU '(UUEncode (FUNCTION UU.TEDIT-INCLUDE-ENCODED)
                                                                        "Encode & include a file"
                                                                        (SUBITEMS ("UUDecode"
                                                                                   (FUNCTION UU.DECODE-FROM-TEDIT)
                                                                                   "Decode the text in this
                                                                                   window"]
              (GLOBALVARS TEDIT.DEFAULT.MENU))
        (PROP FILETYPE UUENCODE)))

;; UNIX compatible uuencode & uudecode.
;; encoding

(DEFINEQ

(UUENCODE
  [LAMBDA (FILES INTO-FILE)                                   (* drc%: "16-Mar-87 15:37")
    (RESETLST
        (for FILE in (MKLIST FILES) bind INTO-STREAM first (SETQ INTO-STREAM (OPENSTREAM INTO-FILE 'OUTPUT))
                                                           (RESETSAVE NIL (LIST (FUNCTION CLOSEF?)
                                                                                INTO-STREAM))
           do (UUENCODE-ONE-FILE FILE INTO-STREAM) finally (RETURN (CLOSEF INTO-STREAM))))])

(UUENCODE-ONE-FILE
  [LAMBDA (IN-FILE OUT-FILE-OR-STREAM)                        (* drc%: "16-Mar-87 14:34")
    ;; uuencode IN-FILE to OUT-FILE-OR-STREAM.
    (RESETLST
        (LET [(INS (OPENSTREAM IN-FILE 'INPUT]
             (RESETSAVE NIL (LIST (FUNCTION CLOSEF?)
                                  INS))
             (LET* [(ALREADY-OPEN? (OPENP OUT-FILE-OR-STREAM 'OUTPUT))
                    (OUTS (if ALREADY-OPEN?
                              then (GETSTREAM OUT-FILE-OR-STREAM 'OUTPUT)
                            else (OPENSTREAM OUT-FILE-OR-STREAM 'OUTPUT]
                   (if (NOT ALREADY-OPEN?)
                       then (RESETSAVE NIL (LIST (FUNCTION CLOSEF?)
                                                 OUTS)))
                   (UUENCODE-INTERNAL INS OUTS (NAMEFIELD IN-FILE T))
                   (CLOSEF INS)
```

```
                    (if (NOT ALREADY-OPEN?)
                        then (CLOSEF OUTS)
                      else OUTS))))])
```

(UUENCODE-INTERNAL

```
  [LAMBDA (INS OUTS DECODE-NAME FILE-MODE)                    ; Edited 7-Oct-87 17:22 by drc:
    ;; encode text from INS to OUTS.
    ;; DECODE-NAME is what the file should be called when decoded
    ;; FILE-MODE is the UNIX file mode.  The default is reasonable.
    (LET* [(EOF (GETEOFPTR INS))
           (PADDING (CL:MOD EOF 3))
           (STOP (CL:IF (ZEROP PADDING)
                     EOF
                     (PLUS EOF (- 3 PADDING)))]
          ;; Each 3 bytes are encoded in 4 six-bit chars.
          (from 1 to STOP first ;; Print header
                                (printout OUTS T "begin ")
                                (RESETFORM (RADIX 8)
                                       (printout OUTS (OR FILE-MODE UU.MODE-DEFAULT)))
                                (printout OUTS " " (OR DECODE-NAME (NAMEFIELD (FULLNAME INS)
                                                                             T)))
             bind (STATE _ 1)
                  BYTE BITS (COLUMN _ 0)
             declare (LOCALVARS . T) do [if (ZEROP COLUMN)
                                            then ;; time to start a new line
                                                 (TERPRI OUTS)
                                                 ;; first char represents how much cleartext will be on the line
                                                 (BOUT OUTS (UU.BYTE (IMIN UU.CHARS-PER-LINE (IDIFFERENCE
                                                                                               EOF
                                                                                               (GETFILEPTR INS]
                                        (if (NOT (EOFP INS))
                                            then (SETQ BYTE (BIN INS))
                                          else 0)
                                        (SELECTQ STATE
                                             (1 (BOUT OUTS (UU.BYTE (RSH BYTE 2)))
                                                (SETQ BITS (LOGAND BYTE UU.LAST-TWO-BITS)))
                                             (2 [BOUT OUTS (UU.BYTE (LOGOR (LSH BITS 4)
                                                                           (RSH BYTE 4]
                                                (SETQ BITS (LOGAND BYTE UU.LAST-FOUR-BITS)))
                                             (3 [BOUT OUTS (UU.BYTE (LOGOR (LSH BITS 2)
                                                                           (RSH BYTE 6]
                                                (BOUT OUTS (UU.BYTE (BITCLEAR BYTE UU.FIRST-TWO-BITS))))
                                             (SHOULDNT))
                                        (SETQ STATE (ADD1 (CL:MOD STATE 3)))
                                        (SETQ COLUMN (CL:MOD (ADD1 COLUMN)
                                                             UU.CHARS-PER-LINE))
             finally ;; print footer
                     (printout OUTS T " " T "end" T)))
      OUTS])

)

(RPAQ? UU.MODE-DEFAULT 420)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS UU.MODE-DEFAULT)
)


(DEFMACRO UU.BYTE (BYTE)
   `(IPLUS ,BYTE (CHARCODE SPACE)))

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

(RPAQQ UU.CHARS-PER-LINE 45)

(RPAQQ UU.LAST-TWO-BITS 3)

(RPAQQ UU.LAST-FOUR-BITS 15)

(RPAQQ UU.FIRST-TWO-BITS 192)

(CONSTANTS (UU.CHARS-PER-LINE 45)
       (UU.LAST-TWO-BITS 3)
       (UU.LAST-FOUR-BITS 15)
       (UU.FIRST-TWO-BITS 192))
```

```
)
)
```

;; decoding

```
(DEFINEQ
```

(UUDECODE

```
  [LAMBDA (FILE-OR-STREAM ONLY-ONE-FILE?)                    (* drc%: "22-Mar-87 15:08")
```

;; decode from FILE-OR-STREAM

;; if ONLY-ONE-FILE? is non-NIL then return name of file extracted.

;; if ONLY-ONE-FILE? is NIL then return list of files extracted.

```
    (RESETLST
        (LET* [(ALREADY-OPEN? (OPENP FILE-OR-STREAM 'INPUT))
               (INS (if ALREADY-OPEN?
                        then (GETSTREAM FILE-OR-STREAM 'INPUT)
                      else (OPENSTREAM FILE-OR-STREAM 'INPUT]
              (if (NOT ALREADY-OPEN?)
                  then (RESETSAVE NIL (LIST (FUNCTION CLOSEF)
                                            INS)))
              (if ONLY-ONE-FILE?
                  then (LIST (UUDECODE-INTERNAL INS ONLY-ONE-FILE?))
                else (bind OUT-FILE eachtime (SETQ OUT-FILE (UUDECODE-INTERNAL INS ONLY-ONE-FILE?)) while
                                                                                                  OUT-FILE
                        collect OUT-FILE))))])
```

(UUDECODE-INTERNAL

```
  [LAMBDA (INS ONLY-ONE-FILE?)                               (* drc%: "22-Mar-87 15:06")
    (LET [(OUT-FILE (UUDECODE-BEGIN-LINE INS (NOT ONLY-ONE-FILE?]
         (if OUT-FILE
             then (RESETLST
                      (LET [(OUTS (OPENSTREAM OUT-FILE 'OUTPUT]
                           (RESETSAVE NIL (LIST (FUNCTION CLOSEF?)
                                                OUTS))
```

;; decode the body

```
                           (UUDECODE-BODY INS OUTS)
```

;; read the 'end' line

```
                           (OR (for BYTE in (CONSTANT (CHCON "end")) always (EQ (BIN INS)
                                                                                BYTE))
                               (ERROR OUT-FILE "NO 'end' LINE FOUND"))
```

;; return the name of the decoded file

```
                           (CLOSEF OUTS)))])
```

(UUDECODE-BEGIN-LINE

```
  [LAMBDA (INS NO-ERROR)                                     (* drc%: "16-Mar-87 15:09")
```

;; Scans for the begin line in file.

;; Returns the name of the encoded file.

;; Returns NIL if end of file is reached and NO-ERROR is specified.

```
    (if NO-ERROR
        then (RESETLST
                 [RESETSAVE NIL (LIST (FUNCTION SETFILEINFO)
                                      INS
                                      'ENDOFSTREAMOP
                                      (GETFILEINFO INS 'ENDOFSTREAMOP]
                 [SETFILEINFO INS 'ENDOFSTREAMOP (FUNCTION (LAMBDA (S)
                                                           (RETFROM 'UUDECODE-BEGIN-LINE NIL]
```

;; what? Interlisp? Cryptic? Never.

```
                 (UUDECODE-BEGIN-LINE-INTERNAL INS))
      else (UUDECODE-BEGIN-LINE-INTERNAL INS])
```

(UUDECODE-BEGIN-LINE-INTERNAL

```
  [LAMBDA (INS)                                              ; Edited  7-Oct-87 17:45 by drc:
    (until (for BYTE in (CONSTANT (CHCON "begin ")) always (AND (EQ (\PEEKBIN INS)
                                                                    BYTE)
                                                                (BIN INS)))
       do
```

;; skip to next line

```
          (until (SELCHARQ (BIN INS)
                           ((CR LF)
                            T)
                           NIL))
       bind FILE-NAME finally
```

;; read mode (ignored)

```
                    (RESETFORM (RADIX 8)
                           (OR (SMALLP (RATOM INS UU.READTABLE))
                               (ERROR (FULLNAME INS)
```

```
                                        "BAD 'begin' LINE")))
                              ;; read space
                              (OR (EQ (BIN INS)
                                      (CHARCODE SPACE))
                                  (ERROR (FULLNAME INS)
                                         "BAD 'begin' LINE"))
                              ;; read file name
                              (SETQ FILE-NAME (RSTRING INS UU.READTABLE))
                              ;; read end of line
                              (SELCHARQ (BIN INS)
                                   ((LF CR))
                                   (ERROR (FULLNAME INS)
                                          "BAD 'begin' LINE"))
                              ;; return the file name
                              (RETURN FILE-NAME])
```

(UUDECODE-BODY

```
  [LAMBDA (INS OUTS)                                    ; Edited  7-Oct-87 17:06 by drc:
    ;; The inner decoding loop.
    (bind LINE-LENGTH declare (LOCALVARS . T) eachtime
                                                ;; read length of line
                                                (SETQ LINE-LENGTH (UU.SIXBITS (BIN INS)))
       while (NOT (ZEROP LINE-LENGTH)) do ;; read body of line, decoding to output as we go
                                          (bind (N-CHARS-READ _ 0)
                                                (NIBBLE-N _ 0)
                                                NIBBLE-1 NIBBLE-2 NIBBLE-3 eachtime (SETQ NIBBLE-N
                                                                                           (ADD1 (IMOD NIBBLE-N 4)))
                                             while (ILESSP N-CHARS-READ LINE-LENGTH)
                                             do ;; nibbles here are six-bits
                                                (SELECTQ NIBBLE-N
                                                     (1 (SETQ NIBBLE-1 (UU.SIXBITS (BIN INS))))
                                                     (2 (SETQ NIBBLE-2 (UU.SIXBITS (BIN INS)))
                                                        (BOUT OUTS (LOGOR (UU.LSH NIBBLE-1 2)
                                                                          (RSH NIBBLE-2 4)))
                                                        (add N-CHARS-READ 1))
                                                     (3 (SETQ NIBBLE-3 (UU.SIXBITS (BIN INS)))
                                                        (BOUT OUTS (LOGOR (UU.LSH NIBBLE-2 4)
                                                                          (RSH NIBBLE-3 2)))
                                                        (add N-CHARS-READ 1))
                                                     (4 [BOUT OUTS (LOGOR (UU.LSH NIBBLE-3 6)
                                                                          (UU.SIXBITS (BIN INS]
                                                        (add N-CHARS-READ 1))
                                                     (SHOULDNT))
                                             finally ;; read padding
                                                (OR (EQ NIBBLE-N 1)
                                                    (from NIBBLE-N to 4 do (BIN INS)))
                                                (SELCHARQ (BIN INS)
                                                     ((LF CR))
                                                     (ERROR (FULLNAME INS)
                                                            "LINE TOO LONG - FILE BASHED?")))
       finally (SELCHARQ (BIN INS)
                         ((LF CR))
                         (ERROR (FULLNAME INS)
                                "LINE TOO LONG - FILE BASHED?"])

)

(DEFMACRO UU.SIXBITS (CHAR)
   `(CL:MOD (IDIFFERENCE ,CHAR (CHARCODE SPACE))
            64))

(DEFMACRO UU.LSH (N BITS)
   `(LDB (BYTE 8 0)
         (LSH ,N ,BITS)))

(RPAQ UU.READTABLE (COPYREADTABLE 'ORIG))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS UU.READTABLE)
)

(SETBRK NIL NIL UU.READTABLE)

(SETSEPR (CHARCODE (SPACE CR LF))
```

```
       NIL UU.READTABLE)

(ESCAPE NIL UU.READTABLE)
```

;; TEdit interface

```
(DEFINEQ

(UU.TEDIT-INCLUDE-ENCODED
  [LAMBDA (TEXTSTREAM)                                    (* drc%: "16-Mar-87 19:28")
    (LET ((FILE (TEDIT.GETINPUT TEXTSTREAM "File to encode:")))
         (if FILE
             then (RESETLST
                      (LET* [(TEMP-STREAM (OPENSTREAM '{NODIRCORE} 'BOTH]
                            (RESETSAVE NIL (LIST (FUNCTION CLOSEF?)
                                                 TEMP-STREAM))
                            (UUENCODE-ONE-FILE FILE TEMP-STREAM)
                            (SETFILEPTR TEMP-STREAM 0)
                            (TEDIT.RAW.INCLUDE TEXTSTREAM (UUENCODE-ONE-FILE FILE TEMP-STREAM))))
          else (TEDIT.PROMPTPRINT TEXTSTREAM " [aborted]"])

(UU.DECODE-FROM-TEDIT
  [LAMBDA (TEXTSTREAM)                                    (* drc%: "22-Mar-87 15:02")
    ;; decode from FILE-OR-STREAM
    ;; if ONLY-ONE-FILE? is non-NIL then return name of file extracted.
    ;; if ONLY-ONE-FILE? is NIL then return list of files extracted.
    (bind OUT-FILE LAST-FILE first (SETFILEPTR TEXTSTREAM 0)
                                  (TEDIT.PROMPTPRINT TEXTSTREAM "Decoding ... " T)
       eachtime (SETQ LAST-FILE OUT-FILE)
                (SETQ OUT-FILE (UUDECODE-INTERNAL TEXTSTREAM))
       while OUT-FILE do (AND LAST-FILE (TEDIT.PROMPTPRINT TEXTSTREAM ", "))
                         (TEDIT.PROMPTPRINT TEXTSTREAM OUT-FILE)
       finally (TEDIT.PROMPTPRINT TEXTSTREAM (if LAST-FILE
                                                 then " done."
                                                 else "nothing to decode!"])

)

[AND (FGETD 'TEDIT)
     (TEDIT.ADD.MENUITEM TEDIT.DEFAULT.MENU '(UUEncode (FUNCTION UU.TEDIT-INCLUDE-ENCODED)
                                                       "Encode & include a file"
                                                       (SUBITEMS ("UUDecode" (FUNCTION UU.DECODE-FROM-TEDIT)
                                                                             "Decode the text in this window"]

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS TEDIT.DEFAULT.MENU)
)

(PUTPROPS UUENCODE FILETYPE :COMPILE-FILE)

(PUTPROPS UUENCODE COPYRIGHT ("Douglass Read Cutting" 1987))
```

FUNCTION INDEX

CONSTANT INDEX

MACRO INDEX

VARIABLE INDEX

PROPERTY INDEX