〔公開〕

TR－C－0051

Ｓｙｍｂｏｌｉｃｓ用　日本語ターミナル・エミュレータ

林　　潔
KIYOSHI HAYASHI

１９９０．　８．　１．

ＡＴＲ通信システム研究所

*Symbolics*用　日本語ターミナル・エミュレータ

1．概　要

シンボリックス上のターミナルウィンドウへの日本語入出力機能を持つターミナルタイプの追加のためのパッチファイル。

日本語コードとしてはＪＩＳの入出力とＥＵＣコードの表示機能を有し、表示コードはＪＩＳとＥＵＣを自動判定し表示できる。ただし、日本語入力機能についてはＪＯＫＥＲシステム（TR-C-0027)と組み合わせて使用した場合のみ可能である。

2．利用方法

2.1 システムのロード

シンボリックスのリスプリスナーのコマンド入力で、

```
Command: Load File "clm01:>local>binary>genera-7-2>network>japanese-simulator"
```

と入力することによってシステムがロードされ仕様可能状態になる。

但し、シンボリックス立ち上げ後、ターミナルウィンドウ起動前にロードファイルしておく必要あり。（ディフォルト値の設定都合のため）

2.2 システムの起動

プレビュアーウインドウは、「 Select 」＋「ｔ」のキー入力により起動される。

ここで、受付られるコマンドは、ターミナルウィンドウのコマンドであるので、シンボリックスのマニュアル参照の事。

また、ｕｎｉｘシステムにコネクトすると自動的に日本語ターミナルエミュレータが選択される。この場合のｕｎｉｘの termcap例を付録１に示す。

2.3 フォントサイズ

シンボリックス画面の漢字フォントの大きさを変数 TELNET:*font-size* に :normal または :large を設定する事で変更することができる。ディフォルトは :large 。

但し、フォントサイズを変更すると画面に表示できる文字数が代わるため termcapの設定も変更要（付録１参照）。

3．保　守

シンボリックスのTELNETパッケージの中にパッチを当てているため、オリジナル関数にパッチを当てたところはボールドフェースで明示している。但し、新たに作成した関数はこの限りではない。

以　上

付　録　1　：　termcap サンプル

```
g72|genera72|Symbolics ANSI terminal (Genera 7.2):¥
        :li#50:co#130:¥
        :bs:¥
        :im:ic=¥E[1@:¥
        :al=¥E[1L:dl=¥E[1M:¥
        :dc=¥E[1P:¥
        :cd=¥E[0J:ce=¥E[0K:cl=¥E[1;1H¥E[2J:¥
        :cm=¥E[%i%d;%dH:ho=¥E[1;1H:¥
        :do=¥E[1B:up=¥E[1A:nd=¥E[1C:¥
        :nl=¥E[1B:
g72-l|genera72-large|Symbolics ANSI terminal (Genera 7.2) use large char :¥
        :li#32:co#114:¥
        :bs:¥
        :al=¥E[1L:dl=¥E[1M:¥
        :dc=¥E[1P:¥
        :cd=¥E[0J:ce=¥E[0K:cl=¥E[1;1H¥E[2J:¥
        :cm=¥E[%i%d;%dH:ho=¥E[1;1H:¥
        :do=¥E[1B:up=¥E[1A:nd=¥E[1C:¥
        :nl=¥E[1B:
```

注〕 (setf TELNET:*font-size* ':normal) では　genera72　を使用

(setf TELNET:*font-size* ':large)では　genera72-large　を使用

付　録　２：　ソースリスト

"clm01:>local>binary>genera-7-2>network>japanese-simulator.lisp"

```
;;; -*- Mode: LISP; Base: 8; Syntax: Zetalisp; Patch-File: Yes; Package: TELNET -*-
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;
;;;      Symbolics JIS code terminal simulator 7.2 (on Genera 7.2)
;;;                                    for Ultrix with nemacs-2.1
;;;
;;;             designed by K.Hayashi 1988 SEP 27
;;;
;;;           [ Network address :  hayashi@atr-sw.atr.junet ]
;;;
;;;         Copyright ATR Communication Systems Research Laboratories
;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(SI:BEGIN-PATCH-SECTION)
(SI:PATCH-SECTION-SOURCE-FILE "SYS:NETWORK;TELNET.LISP.1565")
(SI:PATCH-SECTION-ATTRIBUTES
  "-*- Mode: LISP; Package: TELNET; Base: 8; Syntax: Zetalisp -*-")

(defvar *kanji-shift* nil)
(defvar *font-size* ':large)

(defflavor jis-terminal-simulator
        ((region-top-line nil)
         (region-bottom-line nil)
         (keypad-mode :numeric))
        (ansi-terminal-simulator))

(defmethod (:name jis-terminal-simulator) () "japanese")

(DEFMETHOD (:FILTER JIS-TERMINAL-SIMULATOR) (CH)
  (COND ((= CH #O033)
         (SEND SELF ':HANDLE-ESCAPE))
        ((OR (= CH 0) (= CH #O177)) nil)
        ((or (and *kanji-shift* (< #x20 ch)) (< #x7f ch))       ; 漢字コード
         (send self ':handle-kanji-code (logand #x7f ch)))
        (T
         (DEFAULT-TERMINAL-SIMULATOR CH))))


(defmethod  (:handle-kanji-code jis-terminal-simulator) (ch)
  (send output-stream :force-output)
  (let* ((ch2 (logand #x7f (send input-stream :tyi)))
         (ch2t (if (= ch2 #x1b) (let()(send self ':handle-escape) (send input-stream :tyi))
                   ch2)))
         (jis-to-char (logand #x7f ch) (logand #x7f ch2t))))


(DEFMETHOD (:HANDLE-ESCAPE-COMMAND JIS-TERMINAL-SIMULATOR) (CSI-P CH &REST PARAMS)
  (IF (NOT CSI-P)
      (SELECTOR CH CHAR=
        (#/$
         (LET* ((X (SEND INPUT-STREAM ':TYI)))  ; シフト イン
           (COND ((OR (= X (CHAR-CODE #/B))(= X (CHAR-CODE #/@)))
                  (SETQ *KANJI-SHIFT* T)))))
        (#/(
         (LET* ((X (SEND INPUT-STREAM ':TYI)))  ; シフト アウト
           (COND ((OR (= X (CHAR-CODE #/B))(= X (CHAR-CODE #/H))(= X (CHAR-CODE #/J)))
                  (SETQ *KANJI-SHIFT* NIL)))))
        (OTHERWISE
          (SEND SELF :HANDLE-OFFICIAL-ESCAPE-COMMAND CH)))
      (SELECTOR CH CHAR=
        (#/?      ;Extended parameters to (re)set mode.  Just ignore them.
         (SEND SELF :PARSE-PARAMETERS))
        (#/r      ;DECSTBM set top and bottom margins
         (SETQ REGION-TOP-LINE (IF (FIRST PARAMS) (1- (FIRST PARAMS)) 0)
               REGION-BOTTOM-LINE (IF (SECOND PARAMS) (SECOND PARAMS) 0))
         (SEND *VIEWPORT-STREAM* :HOME-CURSOR))
        (OTHERWISE
          (LEXPR-SEND SELF ':HANDLE-OFFICIAL-ESCAPE-CSI-COMMAND CH PARAMS)))
      ))


(ADD-TERMINAL-SIMULATOR-TYPE 'JIS-TERMINAL-SIMULATOR)
```

```
;;;    Set Default Terminal Simulator = JIS Terminal  & Initialize routine
(DEFCONST *NVT-OVERSTRIKE-DEFAULT* NIL
  "Controls whether overstriking is permitted by default.")


;;; The unix supdup server ignores the overstriking option.
(DEFCONST *SYSTEM-TYPES-THAT-REFUSE-OVERSTRIKING* '(:ULTRIX :UNIX :UNIX42 :VMS :VMS4 :VMS4.4)
)


(DEFVAR *HOST-DEFAULT-TERMINAL-SIMULATORS*
  '((:VMS VT100-TERMINAL-SIMULATOR) (:VMS4 VT100-TERMINAL-SIMULATOR)
    (:VMS4.4 VT100-TERMINAL-SIMULATOR)
    (:ULTRIX JIS-TERMINAL-SIMULATOR)
    (:UNIX JIS-TERMINAL-SIMULATOR)(:UNIX42 JIS-TERMINAL-SIMULATOR)
    (OTHERWISE AMBASSADOR-TERMINAL-SIMULATOR)))


(DEFMETHOD (:TYPEOUT-TOP-LEVEL NVT-WINDOW) ()
  (LET ((*TERMINAL-STREAM* SELF)
        (*VIEWPORT-STREAM* VIEWPORT-STREAM)
        (*TERMINAL-SCREEN* SELF #|(SEND SELF :GET-PANE 'TERMINAL-SCREEN)|#))
    (WHEN (NULL NETWORK-STREAM)
      (PROCESS-DISABLE TYPEOUT-PROCESS)
      (PROCESS-ALLOW-SCHEDULE))
    (CONDITION-CASE (ERROR)
        (LOOP DOING
          (LET ((LAST-OUTPUT-PROCESS-TICK OUTPUT-PROCESS-TICK)
                (FAST-FILTERS (LOOP FOR FILTER FIRST TYPEOUT-FILTERS
                                    THEN (SEND FILTER :INPUT-STREAM)
                                    UNTIL (EQ FILTER NETWORK-STREAM)
                                    AS FAST-FILTER = (SEND-IF-HANDLES FILTER :FAST-FILTER-TES
T)
                                    WHEN (NULL FAST-FILTER) RETURN NIL
                                    UNLESS (EQ FAST-FILTER T) COLLECT FAST-FILTER)))
            (WHEN (NOT (SEND NETWORK-STREAM :OPERATION-HANDLED-P :READ-INPUT-BUFFER))
              (SETQ FAST-FILTERS NIL))
            (DW:WITH-OUTPUT-RECORDING-ENABLED (*TERMINAL-SCREEN* RECORD-OUTPUT-P)
              ;; Since the only mouse commands are those that require the control key down,
              ;; there's no need on getting an immediate update every time we do a
              ;; :CLEAR-CHAR.
              (TV:DELAYING-MOUSE-WAKEUP
                (IF (NULL FAST-FILTERS)
                    (LOOP WHILE (= LAST-OUTPUT-PROCESS-TICK OUTPUT-PROCESS-TICK)
                          DOING
                      (LOOP FOR CH = (SEND TYPEOUT-FILTERS :TYI T)
                                   THEN (SEND TYPEOUT-FILTERS :TYI-NO-HANG T)
                            WHILE CH
                            DO (SEND *TERMINAL-SCREEN* :TYO CH))
                      (SEND *TERMINAL-SCREEN* :FORCE-OUTPUT))
                  (STACK-LET ((CHAR-BUFFER (MAKE-ARRAY 100. :TYPE 'ART-STRING
                                                            :FILL-POINTER 0)))
                    (LOOP WHILE (= LAST-OUTPUT-PROCESS-TICK OUTPUT-PROCESS-TICK)
                          DOING
                      (SETF (FILL-POINTER CHAR-BUFFER) 0)
                      (LET ((SLOW-P NIL))
                        (MULTIPLE-VALUE-BIND (BUFFER START END)
                            (SEND NETWORK-STREAM :READ-INPUT-BUFFER T)
                          (let ((string "")(first-byte t)(ch1 0))
                            (LOOP FOR INDEX FROM START BELOW END
                                  AS CH = (AREF BUFFER INDEX)
                                  WHEN (LOOP FOR FAST-FILTER IN FAST-FILTERS
                                             THEREIS (FUNCALL FAST-FILTER CH))
                                    DO (SETQ SLOW-P T)
                                       (LOOP-FINISH)
                                  ELSE DO (if *kanji-shift*
                                              (if first-byte
                                                  (setq first-byte nil ch1 ch)
                                                  (setf string (format nil "~a~a" string
                                                                       (jis-to-char ch1 ch)))
                                                  (setq first-byte t))
                                              (ARRAY-PUSH-EXTEND CHAR-BUFFER (CODE-CHAR CH)
))
                                  FINALLY (WHEN (> INDEX START)
                                            (if *kanji-shift*
                                                (send *terminal-screen*
                                                      :jis-string-out string)
```

```
                                                    (SEND *TERMINAL-SCREEN*
                                                          :STRING-OUT CHAR-BUFFER))
                                                  (SEND NETWORK-STREAM :ADVANCE-INPUT-BUFFER
                                                        INDEX)))))
                                (WHEN SLOW-P
                                  (LET ((CH (SEND TYPEOUT-FILTERS :TYI T)))
                                    (WHEN CH (SEND *TERMINAL-SCREEN* :TYO CH)))))))))))))
        ((NETWORK-ERROR END-OF-FILE)
        (SEND *TERMINAL-SCREEN* :FORCE-KBD-INPUT '(:ERROR ,ERROR))
        (PROCESS-DISABLE TYPEOUT-PROCESS)))))


(DEFWHOPPER (:TYO TERMINAL-SCREEN) (CH)
    (TYO-IF-WALLPAPERING CH)
    (IF (NOT (GRAPHIC-CHAR-P CH))
        (CONTINUE-WHOPPER CH)
        (if *kanji-shift*
            (multiple-value-bind (scroll width height)
                (if (equal *font-size* ':normal)
                    (values 1 16. 16.)
                    (values 2 18. 20.))
              (multiple-value-bind (x y) (send self :read-cursorpos)
                (send self :draw-rectangle width height x (1- y) tv:erase-aluf)
                (send self :draw-char ch x (+ y scroll) tv:alu-seta)
                (send self :set-cursorpos (+ x width) y)
                ))
            (UNLESS OVERSTRIKE-P
              (SEND SELF :CLEAR-CHAR))
            (WITH-CHARACTER-ATTRIBUTE (CONTINUE-WHOPPER CH)))
        ))



(defun jistochar (code mode)
  (selector code =
    (#x2141 #\～)
    (#x2126 #\・)
    (#x212e (if mode #\‘ #\`))
    (#x2131 (if mode #\~ #\¯))
    (#x222a #\→)
    (#x222b #\←)
    (#x222c #\↑)
    (#x222d #\↓)
    (otherwise (japanese:jis-to-char code mode))))

(defun jis-to-char (ch1 ch2)
  (let ((char (jistochar (+ (* #x100 ch1) ch2) nil)))
    (if char char #\=)))

(defmethod (:jis-string-out terminal-screen) (string &optional (from 0) to)
  (when wallpapering (wallpaper-operation :string-out string from to))
  (multiple-value-bind (scroll width height)
      (if (equal *font-size* ':normal)
          (values 1 16. 16.)
          (values 2 18. 20.))
    (multiple-value-bind (x y)(send self :read-cursorpos)
      (let ((string-width (* width (string-length string)))
            (sc-y (+ y scroll)))
        (send self :draw-rectangle string-width height x (1- y) tv:erase-aluf)
        (do ((dx x (+ dx width))(i 0 (1+ i))) ((≥ i (string-length string)))
          (send self :draw-char (aref string i) dx sc-y tv:alu-seta))
        (send self :set-cursorpos (+ x string-width) y))))))



(defmethod (:clear-rest-of-window terminal-screen :after) (&optional ignore)
  (send self :clear-history))

(defmethod (:select terminal-screen :after) (&optional ignore)
  (setq *kanji-shift* nil)
  (send self ':set-more-p nil)
  (send self ':set-auto-line-height nil)
  (send (cadr (send self ':blinker-list)) ':set-visibility ':blink)
  (if (equal *font-size* ':normal)
      (let ()
        (send self :set-default-style '(:fix :roman :normal))
        (send self :set-line-height-from-styles '(:fix :roman :normal)))
      (send self :set-default-style '(:fix :roman :large))
```

```
      (send self :set-line-height-from-styles '(:swiss :roman :very-large))))


;;;   KANJI shift control for key-in on TELNET

(DEFMETHOD (:FILTER TELNET-FILTER) (CH)
  (COND ((LISTP CH)
         CH)
        ((CL:MEMBER CH '(#\SCROLL #\META-SCROLL) :TEST #'CHAR-EQUAL)
         (CP::SCROLL-WINDOW-COMMAND-INTERNAL :SCREEN (IF (CHAR-EQUAL CH #\SCROLL) +1 -1) :Y
                                             *TERMINAL-STREAM*)
         NIL)
        (T
         (LET ((CODE (CHAR-CODE CH)))
           (WHEN (CHAR-BIT CH :CONTROL)
             (SETQ CODE (LDB (BYTE 5 0) CODE))) ;Controlify
           (WHEN (and (< 200 code 600)(≠ code 202))     ; 漢字用 patch
             (SETQ CODE (AREF *TELNET-KEYS* (- CODE #O200))))
           (COND ((OR (NULL CODE) (LISTP CODE))
                  CODE)
                 ((≤ 600 code)          ;;;  漢 字
                  (send output-stream ':tyo #x1b)       ; esc
                  (send output-stream ':tyo #x24)       ; $
                  (send output-stream ':tyo #x42)       ; B
                  (let ((jis-code (japanese:char-to-jis (code-char code) nil)))
                    (multiple-value-bind (uch lch)
                        (floor (if jis-code jis-code #x222e) 400)
                      (send output-stream ':tyo uch)
                      (send output-stream ':tyo lch)))
                  (send output-stream ':tyo #x1b)       ; esc
                  (send output-stream ':tyo #x28)       ; (
                  #x42)                                 ; B
                 ((= code 202)          ;;;  CLEAR-INPUT
                  (send output-stream ':tyo #x01)       ; c-a
                  #x0b)                                 ; c-k
                 (T                     ;;;  ASCII
                  (WHEN (CHAR-BIT CH :META)
                    (send output-stream ':tyo #x1b)     ; meta code
                    (WHEN (OR (≤ (CHAR-CODE #/A) CODE (CHAR-CODE #/Z))
                              (≤ (CHAR-CODE #/a) CODE (CHAR-CODE #/z)))
                      (SETQ CODE (LOGXOR CODE #O40)))   ;Flip case (m-A ≠ m-a).
                    (SETQ CODE (LOGIOR CODE #O200)))
                  CODE))))))


(DEFMETHOD (:DELETE-LINE TERMINAL-SCREEN) (&OPTIONAL (LINES 1))
  (MULTIPLE-VALUE-BIND (NIL Y) (SEND SELF :READ-CURSORPOS)
    (LET* ((LH (SEND SELF :LINE-HEIGHT))
           (DY (+ (* LH LINES) Y)))
      (SEND SELF :BITBLT-WITHIN-SHEET TV:ALU-SETA 1065. (- 728. DY) 0 DY 0 Y))))


(DEFMETHOD (:INSERT-LINE TERMINAL-SCREEN) (&OPTIONAL (LINES 1))
  (MULTIPLE-VALUE-BIND (NIL Y) (SEND SELF :READ-CURSORPOS)
    (LET* ((LH (SEND SELF :LINE-HEIGHT))
           (DY (+ (* LH LINES) Y)))
      (SEND SELF :BITBLT-WITHIN-SHEET TV:ALU-SETA 1065. (- DY 728.) 0 Y 0 DY)
     (SEND SELF :CLEAR-BETWEEN-CURSORPOSES 0 Y 0 DY)
      )))


;;;
;;;   end of file
;;;
```