To: Distribution

From: Robert S. Coren

Date: 07/16/76

Subject: Increased User Control Over Terminal Behavior

INTRODUCTION

It appears useful -- and there is certainly appreciable popular demand -- to allow a user more control over the behavior of his/her terminal than is currently available. In particular, the only parameters the user can alter at present are terminal type and modes. It is proposed that a mechanism be provided for a user to alter any of the following:

output delay timings

input editing characters (erase, kill)

escape sequences

translation tables

"special" characters

At present, input editing characters are fixed system-wide; the other parameters are a function of terminal type, and the only way to change any of them is to change them all by setting one's terminal type. Besides being undesirable in itself, this introduces the problem of requiring a multitude of terminal types to allow for the slight variations among existing (and potential) terminals. The related issue of installation-definable terminal types and possible expansion of the initial modes table will be discussed in a future MTB.

IMPLEMENTATION

Control operations will be provided in the hardcore tty DIM to substitute each of the relevant tables used by the hardcore on a per-channel basis; in addition, control arguments will be added to the set_tty command for modifying and printing delay timings and input editing characters. Modification of the other parameters mentioned above will be restricted to the substitution of entire tables, and will accordingly require knowledge of the formats of the relevant tables; we propose to document the control operations in the SWG rather than in the MPM Subroutines.

Whenever the user changes his/her terminal type, the default tables for the new terminal type are adopted; thus if any special tables have been substituted, changing the terminal type undoes the effect of the substitution.

A potential problem arises when the Initializer temporarily seizes a user process's terminal to write warning messages, since user-ring pointers set in the user's process are meaningless to the Initializer. Therefore tty_write must recognize this situation and use the default tables for the terminal type when the caller is not the terminal's user process; the resulting output may be unintelligible, but that is preferable to having the Initializer take faults in ring zero. there is a proposal in the works that will alleviate this problem by having the Initializer use the send_message facility whenever possible rather than seizing the terminal.

The remainder of this MTB consists of SWG documentation of the proposed new control operations, MPM documentation of the proposed new control arguments to set_tty, and a brief summary of the processing of input and output characters intended to indicate how the various tables are used.

Control Operations

For the control operations described below whose names begin with "set", with the exception of set_editing_chars, the tty_ DIM does not copy the user's table, but simply copies the pointer supplied by the user. The user must therefore neither destroy nor modify such a table after making one of these calls.

set_delay
: sets the numbers of delay characters associated with the output of carriage motion characters. The info_ptr points to the following structure:

```
dcl 1 delay based aligned,
      2 version fixed bin,
      2 default fixed bin,
      2 vert_nl fixed bin,
      2 horz_nl fixed bin,
      2 const_tab fixed bin,
      2 var_tab fixed bin,
      2 backspace fixed bin,
      2 vt_ff fixed bin;
```

version
: is the version number of the structure. It must be 1.

default
: indicates, if nonzero, that the default values for the current terminal type and baud rate are to be used. If it is not zero, the remainder of the structure is ignored.

vert_nl
: is the number of delay characters to be output for all newlines to allow for the linefeed. If it is negative, it is the complement of the minimum number of characters that must be transmitted between two linefeeds (for a device such as a TermiNet 1200).

horz_nl
: is a factor used to determine the number of delays to be added for the carriage

return portion of a newline, depending on column position. The formula for calculating the number of delay characters to be output following a newline is:

```
ndelays = vert_nl + (horz_nl*column)/512
```

const_tab	is the constant portion of the number of delays associated with any horizontal tab character.
var_tab	is a factor used to determine the number of additional delays associated with a horizontal tab depending on the number of columns traversed. The formula for calculating the number of delays to be output following a horizontal tab is: `ndelays = const_tab + (var_tab*n_columns)/512`
backspace	is the number of delays to be output following a backspace character. If it is negative, it is the complement of the number of delays to be output with the first backspace of a series only (or a single backspace). This is for terminals such as the TermiNet 300 which need delays to allow for hammer recovery in case of overstrikes, but do not require delays for the carriage motion associated with the backspace itself.
vt_ff	is the number of delays to be output following a vertical tab or form-feed.

get_delay	is used to find out what delay values are currently in effect. The info_ptr points to the structure described for set_delay (above) which is filled in as a result of the call.
set_editing_chars	changes the characters used for editing input. The info_ptr points to the following structure:

```
dcl 1 editing_chars aligned,
      2 version fixed bin,
      2 erase char (1) unaligned,
      2 kill char (1) unaligned;
```

version — is the version number of this structure. It must be 2. (Version 1 is used by the Network software.)

erase — is the erase character.

kill — is the kill character.

Note: The following rules apply to editing characters:

1. The two editing characters may not be the same.
2. No carriage-movement character (carriage return, newline, horizontal tab, backspace, vertical tab, or formfeed) may be used for either of the editing functions.
3. NUL and space may not be used for either editing function.
4. If either of the editing characters is an ASCII control character, it will not have the desired effect unless ctl_char mode is on.

get_editing_chars — is used to find out what input editing characters are in effect. The info_ptr points to the structure described above for set_editing_chars, which is filled in as a result of the call.

set_input_translation — provides a table to be used for translation of terminal input to ASCII. The info_ptr points to a structure of the following form:

```
dcl 1 translation_info aligned,
      2 version fixed bin,
      2 default fixed bin,
      2 table aligned,
      3 entries (0:127) char (1) unaligned;
```

version
: is the version number of the structure. It must be 1.

default
: indicates, if nonzero, that the default table for the current terminal type is to be used. If it is not zero, the remainder of the structure is ignored.

The table is indexed by the value of a typed input character, and the corresponding entry contains the ASCII character resulting from the translation. If the info_ptr is null, no translation is to be done.

Note: In the case of a terminal that inputs 6-bit characters and case-shift characters, the first 64 characters of the table correspond to characters in lower shift, and the last 64 to characters in upper shift.

set_output_translation
: provides a table to be used for translating ASCII characters to the code to be sent to the terminal. The info_ptr points to a structure like that described for set_input_translation (above). The table is indexed by the value of each ASCII character, and the corresponding entry contains the character to be output. If the info_ptr is null, no translation is to be done.

Note: For a terminal that expects 6-bit characters and case-shift characters, the 100(8) bit should be turned on in each entry in the table for a character that requires upper shift.

set_input_conversion
: provides a table to be used in

converting input to identify escape sequences and certain special characters. The info_ptr points to a structure of the following form:

```
dcl 1 conversion_info aligned,
    2 version fixed bin,
    2 default fixed bin,
    2 table aligned,
    3 entries (0:127) fixed bin (8) unaligned;
```

version	is as above.
default	is as above.

The table is indexed by the ASCII value of each input character (after translation, if any), and the corresponding entry contains one of the following values:

0 -- ordinary character

1 -- break character

2 -- escape character

3 -- character to be thrown away

4 -- form-feed character (to be thrown away if page-length is nonzero)

set_output_conversion
provides a table to used in formatting output to identify certain kinds of special characters. the info_ptr points to a structure like that described for set_input_conversion (above). The table is indexed by each ASCII output character (before translation, if any), and the corresponding entry contains one of the following values:

0 -- ordinary character

1 -- new-line

2 -- carriage return

3 -- horizontal tab

4 -- backspace

5 -- vertical tab

6 -- form-feed

7 -- character requiring octal escape

8 -- red ribbon shift

9 -- black ribbon shift

10 -- character does not change the column position

11 -- this character together with the following one do not change the column position (used for hardware escape sequences)

17 or greater -- a character requiring a special escape sequence. The indicator value is the index into the escape table of the sequence to be used, plus 16.

get_input_translation
get_output_translation
get_input_conversion
get_output_conversion
These orders are used to obtain the current contents of the specified table. The info_ptr points to a structure like the one described for the corresponding "set" order above, which is filled in as a result of the call. In the case of translation tables, if the specified table does not exist (no translation is required), the status code error_table_$no_table is returned.

set_special provides a table which specifies sequences to be substituted for certain output characters, and characters which are to be interpreted as parts of escape sequences on input. Output sequences are of the following form:

```
dcl 1 c_chars based aligned,
      2 count fixed bin (8) unaligned,
      2 chars (3) char (1) unaligned;
```

count	is the actual length of the sequence in characters (0 <= count <= 3). If count is zero, there is no sequence.
chars	are the characters that make up the sequence.

The info_ptr points to a structure of the following form:

```
dcl 1 special_chars aligned based,
    2 version fixed bin,
    2 default fixed bin,
    2 nl_seq aligned like c_chars,
    2 cr_seq aligned like c_chars,
    2 bs_seq aligned like c_chars,
    2 tab_seq aligned like c_chars,
    2 vt_seq aligned like c_chars,
    2 ff_seq aligned like c_chars,
    2 printer_on aligned like c_chars,
    2 printer_off aligned like c_chars,
    2 red_ribbon_shift aligned like c_chars,
    2 black_ribbon_shift aligned like c_chars,
    2 end_of_page aligned like c_chars,

    2 escape_length fixed bin,
    2 not_edited_escapes (10 refer (escape_length)) like c_chars,
    2 edited_escapes (10 refer (escape_length)) like c_chars,

    2 input_escapes aligned,
      3 len fixed bin (8) unaligned,
      3 str char (1 refer (input_escapes.len)) unaligned,
    2 input_results aligned,
      3 pad bit (9) unaligned,
      3 str  char (1 refer (input_escapes.len)) unaligned;
```

version	is the version number of this structure. It must be 1.

default	is as above.
nl_seq	is the output character sequence to be substituted for a newline character.
cr_seq	is the output character sequence to be substituted for a carriage return character. If count is zero, the appropriate number of backspaces is substituted.
bs_seq	is the output character sequence to be substituted for a backspace character. If count is zero, a carriage return and the appropriate number of blanks are substituted.
tab_seq	is the output character sequence to be substituted for a horizontal tab. If count is zero, the appropriate number of blanks is substituted.
vt_seq	is the output character sequence to be substituted for a vertical tab. If count is zero, no characters are substituted.
ff_seq	is the output character sequence to be substituted for a formfeed. If count is zero, no characters are substituted.
printer_on	is the character sequence to be used to implement the "printer_on" control operation. If count is zero, the function is not performed.
printer_off	is the character sequence to be used to implement the "printer_off" control operation. If count is zero, the function is not performed.
red_ribbon_shift	is the character sequence to be substituted for a red ribbon-shift character. If count is zero, no characters are substituted.
black_ribbon_shift	is the character sequence to be substituted for a black ribbon-shift character. If count is zero, no characters are substituted.

end_of_page	is the character sequence to be printed to indicate that a page of output is full.
escape_length	is the number of output escape sequences in each of the two escape arrays.
not_edited_escapes	is an array of escape sequences to be substituted for particular characters if the terminal is in ""edited" mode. This array is indexed according to the indicator found in the corresponding output conversion table.
edited_escapes	is an array of escape sequences to be used in "edited" mode. It is indexed in the same fashion as not_edited_escapes.
input_escape_length	is the number of characters in each of the strings input_escapes and input_results.
input_escapes	is a string of characters each of which forms an escape sequence when preceded by an escape character.
input_results	is a string of characters each of which is to replace the escape sequence consisting of an escape character and the character occupying the corresponding position in input_escapes (above).

Note: nl_seq.count should generally be nonzero, as should either cr_seq.count or bs_seq.count.

get_special	is used to obtain the contents of the special_chars table currently in use. The info_ptr points to the following structure:

```
dcl 1 get_special_info aligned,
    2 area_ptr ptr,
    2 table_ptr ptr;
```

area_ptr	points to an area in which a copy of the current special_chars table is returned. (Input)
table_ptr	is set to the address of the returned copy of the table. (Output)

Additional_Control_Arguments_to set_tty

-delay values,
-dly values
: sets the delay timings for the terminal according to values, which must be six decimal integers specifying vert_nl, horz_nl, const_tab, var_tab, backspace, and vt_ff, in that order. The meanings of the values are as follows:

vert_nl
: is the number of delay characters to be output for all newlines to allow for the linefeed. If it is negative, it is the complement of the minimum number of characters that must be transmitted between two linefeeds (for a device such as a TermiNet 1200).

horz_nl
: is a factor used to determine the number of delays to be added for the carriage return portion of a newline, depending on column position. The formula for calculating the number of delay characters to be output following a newline is:

```
ndelays = vert_nl + (horz_nl*column)/512
```

const_tab
: is the constant portion of the number of delays associated with any horizontal tab character.

var_tab
: is a factor used to determine the number of additional delays associated with a horizontal tab depending on the number of columns traversed. The formula for calculating the number of delays to be output following a horizontal tab is:

```
ndelays = const_tab + (var_tab*n_columns)/512
```

backspace
: is the number of delays to be output following a backspace character. If it is negative, it is the complement of the number of delays to be output with the first backspace of a series only (or a single backspace). This is for terminals such as the TermiNet 300 which need delays to allow for hammer recovery in case of overstrikes, but do not require

delays for the carriage motion associated with the backspace itself.

vt_ff is the number of delays to be output following a vertical tab or form-feed.

-edit edit_chars,
-ed edit_chars
changes the input editing characters to those specified by edit_chars. edit_chars is a 2-character string consisting of the erase character and the kill character, in that order.

-print_delay,
-pr_dly prints the delay timings for the terminal.

-print_edit,
-pr_ed prints the input-editing characters for the terminal.

-all, -a is the equivalent of -print -print_edit -print_delay.

SUMMARY_OF_INPUT_PROCESSING

This is a general overview of the operations performed on an input string by the hardcore tty_ DIM. For a more detailed description, see MTB 262.

1. Translation -- the characters are translated from the terminal's code to ASCII, using the input_translation table. If there is no input_translation table, this step is omitted.

2. Canonicalization -- the input string is rearranged (if necessary) into canonical form as described in MTB 251.

3. Editing -- erase and kill editing is carried out, using the editing_chars string described above.

4. Break_and_escape_processing -- the characters in the input string are looked up in the input_conversion table and treated accordingly. If a character is preceded by an escape character (as determined from the table) it is looked up in the input_escapes array in the special_chars table, and, if found, replaced by the corresponding character from the input_results array.

SUMMARY_OF_OUTPUT_PROCESSING

This is a general overview of the operations performed on an output string by the hardcore tty_ DIM. For a more detailed description, see MTB 234.

1. Capitalization -- lowercase letters are replaced by uppercase for terminals in "capo" mode; uppercase letters are prefixed by escape characters if appropriate.

2. Formatting -- the characters in the output string are looked up in the output_conversion table described above. Carriage-movement characters are replaced by sequences found in the special_chars table, followed by delay characters if so indicated by the delay table. Ribbon-shift characters are likewise replaced by appropriate sequences. Any character whose indicator in the output_conversion table is greater than 16 is the replaced by the (indicator-16)th sequence in either the not_edited_escapes or edited_escapes array in the special_chars table.

3. *Translation* -- the result of step 2 is translated from ASCII to the terminal's code, using the output_translation table. If there is no output_translation table, this step is omitted.