Function Keys as Input to Extended Curses

Special Notices

Information in this document is correct to the best of our knowledge at the time of this writing. Please send feedback by fax to "AIXServ Information" at (512) 823-4009.

Please use this information with care. IBM will not be responsible for damages of any kind resulting from its use. The use of this information is the sole responsibility of the customer and depends on the customer's ability to evaluate and integrate this information into the customer's operational environment.

> **Note:** The information in this document has **not** been verified for AIX 4.1.

About This Document

This document deals with accepting function keys in an Extended Curses programming environment and has been tested against AIX 3.2.5.

Purpose of This Document

Acquisiton of certain key strokes can be problematic because terminals send mutli-byte strings for many of the non-alphanumeric keys. For example, the HFT produces 'ESC' [001q when Function Key 1 is pressed. Adding to the difficulty, the strings produced by the special keys is often different from one terminal type to another. Code development is made much simpler when the Extended Curses library is uses to return to an integer value for each special key pressed independent of the terminal type.

This document includes a program which serves to illustrate the use of the Extended Curses library functions to achieve terminal independence and interpretation of multi-byte special key strings into integer values.

Unprintable or Nondisplayable Characters

This item contains characters that may not print or display correctly on all systems. A flag line has been placed BENEATH each line that contains one or more of those characters. Flag lines can be identified by the characters "@@" in columns 64 and 65. The position of each unprintable or nondisplayable character is identified by one of the following letters contained in the flag line and located directly beneath that character.

```
A - left square bracket          G - logical not
B - right square bracket         H - exclamation point
C - left brace                   I - vertical bar
D - right brace                  J - back slash
E - percent sign                 K - tilde
F - caret (circumflex)           L - back quote (accent)
```

Function Keys

Provided below is a sample program that will accept a function key in the Base, Shift, or Ctrl state and print out the function key pressed. The function keys are recognized as follows:

Function Key	Base	Shift	Ctrl
F1	F1	F13	F25
F2	F2	F14	F26
F3	F3	F15	F27
F4	F4	F16	F28
F5	F5	F17	F29
F6	F6	F18	F30
F7	F7	F19	F31
F8	F8	F20	F32
F9	F9	F21	F33
F10	F10	F22	F34
F11	F11	F23	F35
F12	F12	F24	F36

Subroutine Definitions

The sample program uses the following curses subroutines:

```
getch()
csavetty(TRUE)
extended(FALSE)
cresetty(FALSE)
```

- getch() gets a character from standard input.
- csavetty(TRUE) saves the current extended curses state with the values crmode, noecho, meta, nonl, and keypad so that it can later be reset. This subroutine should be used when the terminal will be used as a simple terminal. The values set by csavetty(TRUE) perform the following functions:
  - crmode turns off canonical processing.
  - noecho turns off echoing of typed characters to the screen.
  - meta allows getch to return an 8-bit character rather than a 7-bit character.
  - nonl turns off the conversion of a new line to carriage return and line feed and also turns off conversion of returns to new lines.

- keypad(TRUE) enables the terminal's keypad which allows the getch subroutine to return a single value representing a function key. The keypad values are defined in /usr/include/cur02.h. The function keys keypad values are defined as follows:

```
#define KEY_F0          0x180       /* function key -         */
#define KEY_F(n)        (KEY_F0+(n))/*    reserve 128 values  */
```

  It is not necessary to use csavetty() in the sample program. Input modes could be set up with their respective subroutines. However, keypad(TRUE) and nonl() must be used in order to retrieve function keys on the IBM 3151.

- extended(FALSE) turns off conversion of multibyte characters to wide characters. Extended should be set to FALSE if an application does input processing with keypad set to TRUE, but also needs to support multibyte characters.

- cresetty(FALSE) restores the terminal to the state saved by the last call to the csavetty() subroutine without showing the data in the curser global variable.

Sample Program

The following program has been tested with the IBM 3151 Model 310, a vt100 emulator, the HFT, and aixterm. Please note that not all terminals support the Base, Shift, and Ctrl states of function keys (e.g. IBM 3151 does not support Ctrl-Function Keys and VT100 only has F1 - F4 keys).

```
/*******************************************************
 * funkey.c - Prints out the function key pressed.      *
 *            Recognizes F1 - F36:                      *
 *              F1 - F12                                *
 *              Shift F1 - Shift F12 (= f13 - f24)      *
 *              Ctrl F1 - Ctrl F12   (= f25 - f36)      *
 *  To Compile: cc -o funkey.c -0 funkey -lcur          *
 *  To Run: funkey                                      *
 *******************************************************/
#include<cur01.h>
#include<cur02.h>
#define ESC 0x1b
main()  {
        C                                                                @@
  int retval;

  /* --- initialize the curses control structures, etc. */
  initscr();

  /* --- preserve the current state of the tty */
  csavetty(TRUE);

  /* --- turn multibyte processing off */
  extended(FALSE);
```

```
  /* --- enable horizontal scrolling */
  scrollok (stdscr,TRUE);

  printw("Enter a function key (ESC to exit) \n");
  refresh();
                                          J                          @@

  /* --- get keys - quit on ESCape - display function key value */

      while( (retval = getch()) != ESC ) {
                               H        C                          @@
         if( retval > KEY_F(0) && retval < KEY_F(37) )
             printw("*** F%d ***\n",retval - KEY_F(0));
                          E     J                                   @@
         else
             switch(retval ) {
                             C                                      @@
               case KEY_UP: printw("*** Up Arrow ***\n"); break;
                                                 J                  @@
               case KEY_DOWN: printw("*** Down Arrow ***\n"); break;
                                                   J                @@
               case KEY_LEFT: printw("*** Left Arrow ***\n"); break;
                                                   J                @@
               case KEY_RIGHT: printw("*** Right Arrow ***\n"); break;
                                                     J              @@
               case KEY_NPAGE: printw("*** Next Page ***\n"); break;
                                                   J                @@
               case KEY_PPAGE: printw("*** Previous Page ***\n"); break;
                                                     J              @@
               case KEY_HOME: printw("*** Home ***\n"); break;
                                                 J                  @@
               case KEY_END: printw("*** End ***\n"); break;
                                               J                    @@
               case KEY_DELETE: printw("*** Delete ***\n"); break;
                                                 J                  @@
               case KEY_IC: printw("*** Insert ***\n"); break;
                                                 J                  @@
               default: printw("Some other character\n"); break;
                                                   J                @@
               }
               D                                                    @@
           refresh();
           }
           D                                                        @@
  /* --- reset the tty to its original state */
  cresetty(FALSE);
  endwin();
}
D                                                                   @@
```

References

Info Explorer, "General Programming Concepts", Chapter 2, "Curses and Extended Curses".

Strang, John. "Programming with curses", O'Reilly & Associates, Inc.

Reader's Comments

Please fax this form to (512) 823-4009, attention "AIXServ Information". You may also e-mail comments to: elizabet@austin.ibm.com. These comments should include the same customer information requested below.

Use this form to tell us what you think about this document. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

If you need technical assistance, contact your local branch office, point of sale, or 1-800-CALL-AIX (for information about support offerings). These services may be billable. Faxes on a variety of subjects may be ordered free of charge from 1-800-IBM-4FAX. Outside the U.S. call 415-855-4329 using a fax machine phone.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Note: If you have a problem report or item number, supplying that number may help us determine why a procedure did or did not work in your specific situation.

Problem Report or Item #: Branch Office or Customer #:

Be sure to print your name and fax number below if you would like a reply:

Name: Fax Number:

END OF DOCUMENT (fkeys.curses.325.zap, 4FAX 3754)