09/18/95

Drawing Boxes in a Curses Environment with AIX 3.2

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 contains information on drawing boxes using curses calls in AIX 3.2. C programs and shell scripts are used to illustrate curses calls.

Curses Libraries

AIX 3.2 provides two curses libraries. The base curses library (libcurses) is similar to a very early AT&T curses library. It does not include support for box characters. The extended curses library (libcur) is IBMs extension of base curses. It is not compatible with any release of AT&T curses. Extended curses does include support for box characters.

AIX 4.1 includes a curses library consistent with a current release of the AT&T curses library. This document discusses ONLY AIX 3.2 and its extended curses library.

About the Extended Curses Library

Included in the extended curses library are the following subroutines; box, cbox, cboxalt, drawbox, drawboxalt, fullbox, superbox, and superbox1.

The box, cbox, cboxalt, drawbox, drawboxalt, fullbox, superbox and superbox1 subroutines draw a box IN or AROUND a specified window. The box, cbox, cboxalt, and fullbox sub- routines draw a box AROUND the window specified.

Terminal Control Definition Files

The strings of characters necessary to control cursor movement, terminal characteristics, etc. for all supported terminals are found in a family of files found in /usr/lib/terminfo. An examination of this directory will suggest that these terminal definition files are grouped generally by manufacturer and have a .ti suffix. These .ti files are flat (ascii) files in which terminal control attribute names are matched to character sequences. For example the terminal control attribute name 'clear' is defined as being to \EK (the escape character (0x2b) and the letter K) for the ibm3101 terminal.

These flat files are not used directly. For speed of access, they are digested into binary equivalents. This digestion is accomplished with the 'tic' command. Therefore any changes made to a .ti file would not take effect until the .ti file was re-digested with the tic command.

Accessing Terminal Control Sequences

Terminal control sequences can be accessed at both the shell level and from within a 'C' program. An example follows which illustrates the use of the 'tput' command for accessing terminal control sequences from the shell.

A second example will illustrate the use of the extended curses subroutine calls to the same effect.

Box Characters And Terminal Dependencies

It is entirely possible that the methods defined herein will not work in a particular setting. In order for ANY terminal to draw box characters, that terminal must have access to a font in which box characters are defined. Further, the font in which box characters are defined may vary from one terminal type to another. Conventions exist which increase the probability of finding the box character set in a predictable location, but there is no standard which fixes it. That is to say that, if you find that box characters are not available at your terminal, you do not necessarily have a defect of any sort.

Accessing Terminal Control Sequences from the Shell

The 'tput' command is used at the shell level to access a terminal control sequence. For example, 'tput clear' results in the following:

- Your TERM variable is examined.
- The result is used to locate the digested version of the aforementioned .ti file in that digested file.
- The 'clear' attribute is located in this file.
- The sequence of characters matched to the 'clear' attribute are then extracted and written to the terminal.

The 'tput' command will be used repeatedly throughout this document.

Does Your Terminal Support Box Characters?

In the same way that a .ti file matches a string of characters to the 'clear' attribute name, it also matches strings of characters to the 'box1' attribute and attributes whose names are 'font0', 'font1', etc.

We will select various fonts using the fontX attribute then write the string associated with the 'box1' attribute until we learn which font (if any) contains a set of characters suitable for drawing boxes.

At the command line, type in

```
tput font0; tput box1
```

If you see a string of non-box characters, then the font selected with the 'tput font0' command does not contain the necessary characters. Type the above command replacing 'font0' with 'font1', then 'font2', etc., until you see that the box characters are found, or you get an error:

```
'tput: unknown capname'
```

If you got to the error before finding a set of box characters, then the fonts that your terminal knows about do not include the necessary characters.

For the purpose of this discussion, we will assume that 'tput font0' results in the selection of a normal alpha-numeric character set and that 'tput font1' gives access to a set of box characters.

Drawing Box Characters from a Shell Script

This routine gets the characters with the "tput box1" command and then changes the terminal into line drawing font and outputs the characters.

```
#!/bin/ksh

GPH_OFF=$(tput font0)         # select the alpha-num char set
GPH_ON=$(tput font1)          # select the font with box chars

GRPH=$(tput box1)             # get the box chars

# Extract the individual characters into shell variables
TL=$(echo $GRPH | cut -c1)    # top left
HB=$(echo $GRPH | cut -c2)    # horiz bar
TR=$(echo $GRPH | cut -c3)    # top right
VB=$(echo $GRPH | cut -c4)    # vert bar
BR=$(echo $GRPH | cut -c5)    # bottom right
BL=$(echo $GRPH | cut -c6)    # bottom left
TT=$(echo $GRPH | cut -c7)    # top tee
RT=$(echo $GRPH | cut -c8)    # right tee
BT=$(echo $GRPH | cut -c9)    # bottom tee
LT=$(echo $GRPH | cut -c10)   # left tee
CT=$(echo $GRPH | cut -c11)   # center cross

# Put the terminal into line drawing mode
echo "$GPH_ON"

# Output the box characters to the screen
echo "$TL$HB$HB$HB$HB$HB$HB$HB$HB$HB$TR"
echo "$VB          $VB"
echo "$VB          $VB"
echo "$VB          $VB"
echo "$VB          $VB"
echo "$VB          $VB"
echo "$VB          $VB"
echo "$VB          $VB"
echo "$BL$HB$HB$HB$HB$HB$HB$HB$HB$HB$BR"

# Put the terminal back into character mode.
echo "$GPH_OFF"

# End of example script
```

Drawing Box Characters from a C Program

Following a call to the setupterm function, global variables are set up for the names defined in:

```
/usr/include/term.h.
```

The applicable variables compared to the ones used in the shell script above are:

- font_0 (character mode)
- font_1 (line drawing mode)

- box_chars_1

Box Character Differences in Line Drawing and Normal Mode

The following program shows the difference between dis- playing the box characters in both line drawing mode and in normal mode. The program does not attempt to draw a box with the characters, but simply displays them. The effect is similar to typing 'tput fontX' at the shell command line.

This program uses the base curses library. To compile it, type:

```
cc -0  <filename>.c -o <filename> -l curses
```

```
#include <curses.h>
#include <term.h>
main() {
    int err;
    int xcode;
    char capfff40“;
    setupterm(0,1,&err);
    printf ("This is it,font1 %s\n",font_1);
    printf ("Normal chars in line font\n");
    printf ("box1 = %s\n",box_chars_1);
    printf ("%s this is font0\n",font_0);
    printf ("back to normal\n");
    printf ("box1 = %s\n",box_chars_1);
    exit (0);
}
```

Boxes Drawn with Superbox1 or CBOX

The following sample program shows a simple box drawn with the superbox1 or cbox call depending which line is commented out.

This program uses the extended curses library. To compile it, type:

```
cc -0  <filename>.c -o <filename> -l cur
```

```
#include <cur00.h>
#include <cur03.h>
#define A_REVERSE -1

main () {
    if( initscr() == NULL ) {
        printf("%s\n","Error in screen init");
        sleep(1);
        exit( 1 );
        }
    accessmenu ();
    endwin();
    exit(1);
}

accessmenu() {
    WINDOW *menu;
    menu = newwin(8,16,0,0);
    cbox(menu);
    wrefresh(menu);
    mvwgetch(menu, 5, 5);
}
```

To see a variation, replace the cbox(menu) line with superbox1(menu, 0, 0, 8, 16, 32, 32, 32, 32, 32, 32, A_REVERSE);

C Routine Using Drawbox

Another example of a C routine to draw a box using drawbox. This program displays characters used to draw boxes similar to InfoExplorer using IBM3151 and vt100 screen types.

This program uses the extended curses library. To compile it, type:

```
cc -O  <filename>.c -o <filename> -l cur
```

```
#include <cur00.h>
main() {
    initscr();
    refresh();
    mvaddstr(1, 1, "Here's a string");
    refresh();
    chgat(5,FONT0);
    drawbox(stdscr, 2, 2, 8, 20);
    refresh();
    mvaddstr(3, 3, "I'm in the box");
    refresh();
    system("sleep 5");
    wclear(curscr);
    wrefresh(curscr);
    endwin();
}
```

09/18/95

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 (crsbx.32.zap, 4FAX # 3914)