Intro to Curses

Table of Contents

- Curses
- About Curses
- Curses & MS C ver. 5.0
- Curses Programming
- Curses Basics
- Curses Output
- Curses Input
- Curses Effects
- Curses Sample Program
  - 
  - 
  -
- Curses Port Problems
Curses

- developed by Ken Arnold at Berkley, written in C
- provides elementary window and advanced cursor control
- uses the termcap terminfo database for terminal independence under UNIX
- stores current & updated screen images to perform efficient screen manipulations
About Curses

✦ Background Information
  • `/etc/termcap` (terminfo → sys V AT&T) is a database storing specific terminal escape sequences for screen capabilities.
  • Programs written with curses operate independently of the type of terminal the program is executing upon.
  • `curses` accesses the TERM environment variable for a user’s terminal type & searches the termcap database for the specific escape sequences for the terminal’s capabilities.
  • `curses` frees programmers from worrying about what type of terminal their program is executing upon.

✦ Curses Limitations
  • does not provide virtual screens → windows cannot be larger than the physical screen
  • provides no dynamic window management operations for resizing or moving windows

✦ Predefined curses windows
  • `curscr` stores the physical screen contents actual current visible contents of the screen
  • `stdscr` default window representing terminal screen
  • direct access to `curscr` is not allowed, changes are made to `stdscr` & user defined windows
Curses & MS C 5.0

- Curses files for use with MS C 5.0
  - curses.lib  curses library code for the curses package
  - curses.h    header file for calling curses routines
  - Copy or move the curses.lib & curses.h files to the Visual C standard compiler lib & include directories:
    (e.g., \DevStudio\VC\lib)

- Linking with curses under MS C 5.0

Project Type: Win32 Console Application)
(do Not use MS Foundation Classes)

MS C 5.0 Project menu, select Settings . . .
(Select Project entry in the Workspace Window & not a project file.)
Select the Link Tab
add “curses.lib” to the end of the “Object/libraries modules” line

- Program size
  - In C, including curses in a system increases the executable about 10K.
  - curses in C is stored in a code library and only the actual routines accessed in a program are included
  - In Unix: curses also uses the termcap library which must be included on the compile line

Warning! The port of the MS Visual C compiler (ver 5.0) interface to curses is still under development. The information in these slides may require updating. Note: all curses routines may not function exactly as described in standard curses documentation due to the port.
Curses Programming

- Curses program framework

  - `{*** curses initialization routines ***}

    initscr() ;       // initialize curses structures
    nonl() ;         // prevent curses translating newline-return
                    & linefeed
    noecho() ;       // do not echo input
    raw() ;          // turn curses unbuffered input on
    erase() ;        // clear screen
    refresh() ;      // display stANdARDscrEEN

    // *** curses fn calls ***
    // user window creation}
    win = newwin (numrows, numcols, row, col) ;

    mvwaddch (win , y, x, ch) ; /move in window & add char
    touchwin (win) ;        // forces curses to redisplay
    wrefresh (win) ;       // update screen routine

    //
    .  .  .

    // *** curses shutdown routines ***
    noraw() ;          //turn unbuffered mode off
    endwin() ;        //shutdown curses: required call
Curses Basics

- Curses functions/procedures

  - Window and cursor routines are organized in pairs, for example:

    ```
    move ( y , x ) ;  wmove ( win, y , x ) ;
    ```

  - the first routine without the win-dow parameter operates upon the stdscr window. The second routine moves the cursor in the passed win-dow.

  - ( all cursor movement is relative to a specific window, not the entire screen.)

  - curses functions return standard C error codes, where a value of zero indicates normal successful termination and nonzero values denote errors. An integer variable (reterr) must be assigned the return code even if it is not checked.

- Curses Documentation

  - the “curses.h” file gives a list of all the accessible curses routines, constants and a very brief explanation of them.

  - any set of UNIX manuals will contain documentation about curses. Online documentation can be accessed with the “man curses” command, online on the web:

    ```
    http://ei.cs.vt.edu/~cs1704/curses.html
    ```
Curses Output

- Window output

  - after curses routines have been called to write into a window the routine:

    wrefresh(win);

    must be called to inform curses to update the physical screen.

  - if a window’s contents has not changed since the last time it was refreshed another call to wrefresh will not redisplay the window even if another window is overlapping (blocking) it.

  - curses must be forced to throw away its window status information and redisplay the entire window with the routine:

    touchwin(win);

  - Note: touchwin(stdscr); should NOT be performed, instead
  - refresh(); should be used to redisplay the entire screen
Curses Input

- **Raw Input**

  - curses works in unbuffered mode and provides routines for inputting characters `getch(ch)` and strings `getstr(str)`.
  - if buffered input is desired the following sequence must be used:
    ```
    noraw(); // turn on buffered I/O
    cin >> filename; // perform buffered I/O
    raw(); // turn on unbuffered I/O
    ```
  - this sequence toggles between I/O modes.
  - unbuffered mode must be in effect any time curses routines are called.
  - if a program using curses terminates abnormally without restoring buffered I/O, the active process shell may hang and will need to be terminated through another shell or through the OS process manager.
Curses Effects

◆ Window Drawing
  • curses provides a routine for placing a outline around a window:

  ```c
  box (win, verticalChar, horizontalChar);
  ```

  • this routine reduces the useable window size by 2 rows & 2 columns. It is the user’s responsibility not to overwrite the box outline, curses does not check this condition.

◆ Character Drawing
  • curses allows characters to be output with the following attributes:

  ```
  A_STANDOUT A_UNDERLINE A_REVERSE A_BLINK
  A_DIM A_BOLD
  ```

  • the predefined attribute constants above are toggled with the routines:

  ```c
  wattroff (win , attribute);
  ```

  • attributes are terminal specific. If an attribute is requested and it is not available on a given terminal type curses will attempt to substitute an alternate effect.
Curses Sample Program

// curdemo.cpp

/**************************************************************************
Author : D Barnette
Compilation Date : 11-20-96
Operations Performed : text window demo
Algorithms : (see references)
References : "UNIX Curses Explained",
             Goodheart, B., Prentice Hall, 1991
             "Programming with Curses"
             J. Strang, O'Reilly & Assoc.s, 1986
             "Advanced C Programming for Displays"
             Marc J. Rochkind, Prentice Hall, 1988
Parameters & transmission modes : (see individual routines)
Input Assertions : none
Output Assertion : terminal I/O remains in buffered mode
Global Variable : curses standard windows
Side Effects : none
Major Data Structures : none
Calling Modules : application dependent
Called Libraries : curses, strings, standard I/O, standard lib
Timing Constraints : none
Exception Handling : none
Assumptions : none
Version : 1.0
**************************************************************************
Purpose:

This is a demo program for the curses text window interface for the MS C compiler ver. 5.0. This program will place 3 windows on the screen with some data about curses and allow a user to select which window to bring to the forefront.

---------------------------------------------------------------------*/
• // includes consts, typedefs & FN prototypes

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "curses.h"
#include "cborders.h"

// ascii consts for L, M, R, Q, l, m, r, q
const  int leftL = 76 ;
const  int middleM = 77 ;
const  int rightR = 82 ;
const  int quitQ = 81 ;
const  int lefts = 108 ;
const  int middles = 109 ;
const  int rights = 114 ;
const  int quits = 113 ;

//type definitions
typedef int Boolean ;
typedef  WINDOW*  window;

// FN prototypes
void windowleft ( window& wl ) ;
void windowmiddle ( window& wm ) ;
void windowright ( window& wr ) ;
void menu ( window& menuwin ) ;
void errmsg ( window& errwin ) ;
void clearerr ( window& errwin ) ;
void setcolors(window& win) ;
void createwin (window& win, int numlines, int numcols, 
                 int starty, int startx) ;
void wincontrol (window& wl, window& wm, window& wr, 
                 window& menuwin, window& errwin ) ;
```
int main(int argc, char *argv[]) {
    window
    wl, wm, wr, errwin,       // left, middle, right & error windows
    menuwin;                  // menu & input window

    // *** curses initialization routines ***
    initscr();
    setupwin(stdscr);

    nonl();  // do not let curses translate newline - return & linefeed
    noecho(); // *** do not echo input ***
    raw();   // *** turn curses unbuffered input on ***

    erase();  // clear screen
    refresh(); // display stANdARDscrEEN

    // *** define locations of left, middle, right & error windows ***
    createwin(wl, 16, 31, 3, 5);
    createwin(wm, 6, 31, 8, 25);
    createwin(wr, 16, 31, 3, 45);

    errwin = newwin( 1, 80, 23, 0 );
    setupwin(errwin);
// main() continued

// *** put contents into windows ***
windowleft ( wl ) ;
windowmiddle ( wm ) ;
windowright ( wr ) ;

menu ( menuwin ) ;  // *** display menu ***

// *** call curses window display ***
wrefresh ( wl ) ;
wrefresh ( wr ) ;
wrefresh ( wm ) ;

// allow user to view windows
wincontrol (wl, wm, wr, menuwin, errwin) ;

move(23,0);  // ** move cursor to bottom of stdscr **
refresh() ;  // ** display stANdARDscrEEN

noraw() ;  // ** turn unbuffered mode off same as noraw **
endwin() ;  // ** shutdown curses: required call **

return EXIT_SUCCESS ;

}  // end main
void wincontrol (window& wl, window& wm, window& wr, 
   window& menuwin, window& errwin  )
{
    int     key ; // key: char input; reterr: return error
    Boolean  display, keyerr ;

    keyerr = FALSE ;
    display = TRUE ;
    while (display) { // *** bring windows forward until user quits ***
        key = getch() ; // get key char (curses returns int ordinal)
        if (keyerr) {
            // ** clear err msg if illegal key hit previously **
            clearerr( errwin ) ;
            keyerr = FALSE ;
        } // end if
    } // end while
}
Curses Sample Program (cont)

//window control (cont)

switch ( key )
{
    case leftL :  // ** bring left window forward **
        touchwin (wl) ; //forces curses to redisplay
        wrefresh (wl) ;
        break ;
    case lefts :
    case middles :  // ** bring middle window forward **
        touchwin (wm) ; // forces redisplay
        wrefresh (wm) ;
        break ;
    case middleM :
    case middles :
        // ** bring middle window forward **
        touchwin (wm) ; // forces redisplay
        wrefresh (wm) ;
        break ;
    case rightR :
    case rights :  // ** bring right window forward **
        touchwin (wr) ; // forces curses to redisplay
        wrefresh (wr) ;
        break ;
    case quitQ :
    case quits :
        display = FALSE ;
        break ;
    default :
        errmsg(errwin) ;
        keyerr = TRUE ;
        break ; // display error

} // end switch
} // end while

} // end wincontrol
Curses Sample Program (cont)

// left window

void windowleft ( window& wl )
{
    // *** put data into left window ***
    mvwaddstr(wl,1,1,"Curses stores screen images,");
    mvwaddstr(wl,2,1,"only performing actual screen");
    mvwaddstr(wl,3,1,"updates when a screen refresh");
    mvwaddstr(wl,4,1,"is requested by the user.");
    mvwaddstr(wl,5,1,"Curses optimizes the cursor");
    mvwaddstr(wl,6,1,"movement needed to transform");
    mvwaddstr(wl,7,1,"the actual screen, curscr,");
    mvwaddstr(wl,8,1,"into the new updated image.");
    mvwaddstr(wl,9,1,"In order to perform efficient");
    mvwaddstr(wl,10,1,"screen updating, curses code");
    mvwaddstr(wl,11,1,"increases a programs size");
    mvwaddstr(wl,12,1,"by approximately 10K. Curses");
    mvwaddstr(wl,13,1,"uses ");
    wattron(wl, A_BOLD); // win attribute on boldface
    waddstr (wl, "Termcap") ; // add string to window
    woffattr(wl, A_BOLD); // win attribute off boldface
    waddstr ( wl, " database to" ) ;
    mvwaddstr(wl, 14, 1, "store terminal control data.");
} // end windowleft
Curses Sample Program (cont)

◆ // middle window

```c
void windowmiddle ( window& wm )
{
    // *** put data into middle window ***
    mvwaddstr(wm,1,1,"This is a demo of the Curses");
    mvwaddstr(wm,2,1,"text windowing package.");
    mvwaddstr(wm,3,1,"Curses was written by");
    wattron(wm, A_REVERSE); // win attribute on reverse
    waddstr ( wm, "Ken " ); // add string to window
    wmove ( wm,4,1 ); // MOVE Cursor in window
    waddstr( wm, "Arnold" );
    woffattr(wm, A_REVERSE); // win attribute off reverse
    waddstr ( wm, " at Berkley." );
} // end windowmiddle
```

◆ // menu window

```c
void menu ( window& menuwin )
{
    // *** Define & display menu window ***
    menuwin = newwin (3,80,20,0); // define menu win location
    setupwin(menuwin); // set menu colors
    mvwaddstr(menuwin,0,27,"To bring a window forward");
    mvwaddstr(menuwin,1,28,"L-left M-middle R-right");
    mvwaddstr(menuwin,2,35,"Q-quit");
    wrefresh ( menuwin ); // update window
} //end menu
```
void windowright ( window& wr )
{
    // *** put data into right window ***
    mvwaddstr(wr,1,1,"Curses provides two defined");
    mvwaddstr(wr,2,1,"windows: ");
    wattron(wr, A_BOLD); // win attribute on boldface
    waddstr ( wr, "stdscr & curscr.");
    woffattr(wr, A_BOLD); // win attribute off boldface
    mvwaddstr(wr,3,1,"Curscr is a representation of");
    mvwaddstr(wr,4,1,"the physical screen and is");
    mvwaddstr(wr,5,1,"never written to directly.");
    mvwaddstr(wr,6,1,"The former, stdscr, is a full");
    mvwaddstr(wr,7,1,"screen equivalent accessed");
    mvwaddstr(wr,8,1,"whenever multiple windows are");
    mvwaddstr(wr,9,1,"not required. Curses quickly");
    mvwaddstr(wr,10,1,"eliminates redundant screen");
    mvwaddstr(wr,11,1,"updates when bringing curscr");
    mvwaddstr(wr,12,1,"into agreement with stdscr or");
    mvwaddstr(wr,13,1,"user defined windows. Curses");
    mvwaddstr(wr,14,1,"provides over 80 procs & fns.");
} // end windowright
// error window
void errmsg ( window& errwin )
{
    // *** display error msg ***
    setupwin(errwin);
    // win attribute on boldface & reverse
    wattrof(errwin, A_BOLD);
    mvwaddstr(errwin,0,25,"Hit only L, M, R or Q please");
    woffattr(errwin, A_BOLD);
    wrefresh ( errwin );
} // end errmsg

// clear error window
void clearerr ( window& errwin )
{
    // *** erase error msg & redisplay blank error window ***
    werase ( errwin );
    wrefresh ( errwin );
} // end clearerr

// set window colors
void setcolors(window& win)
{
    // ** Set the colors of a window to Gray on Blue
    wattrof(win, B_BLUE | F_GRAY);
} // end setcolors
Curses Sample Program (cont)

- //setup windows
  
  void setupwin(window& win)
  {
    // ** set colors and erase to clear window
    setcolors(win);
    werase(win);
  } // end setupwin

- //turn window attribute effects
  
  void woffattr (window& win, int attrs)
  {
    // ** turn attrs off and reset window colors
    wattroff(win, attrs);
    setcolors(win);
  } // end woffattr

- //create window with double line border
  
  void createwin (window& win, int numlines, int numcols,
  int starty, int startx)
  {
    // ** turn attrs off and reset window colors
    win = newwin ( numlines,numcols,starty,startx ) ;
    setupwin(win); //set colors & clear
    box ( win, C_VD, C_HD ) ;//box with double vert & hort lines
  } // end createwin
Curses Port Problems

- Incomplete Windows

- The above screen the result of failing to erase a window after setting its colors.

- Attribute Lingering

- The above window shows the result of turning on attributes, turning them off and not resetting the colors to cancel all of the attributes set on a window.