UNIX System Programming
Lecture 9: Curses

● Outline
  • Introduction to Curses
  • Basic Curses Routines

● Reference
  • BLP: Chapter 6
  • Curses Reference Pages
Although using the **terminfo** routines gives us very fine control over the terminal it requires writing a lot of low-level code.

The higher level routines in the **curses** library provides a terminal independent way to write full-screen apps.

Linux uses **ncurses** (new curses) which is backwards compatible with the original **curses**.
Lecture 8: Terminals
Introduction to Curses

• **curses** provides basic input and output methods and multiple windows. **ncurses** adds line and box drawing, **color**, pads, panels, menus and forms.

• To use the **ncurses** library include the following header file:
  
  #include <curses.h>
  
  and compile and link using:
  
  cc -o myapp myapp.c -lcurses
A window is a data structure describing a subrectangle of the screen. You can write to and scroll windows. A screen is a subset of windows as large as the terminal screen.

stdscr and curscr are predefined windows. The curscr is an internal structure that is an image of what the terminal looks like. Programs requiring only a single window write output to stdscr. A call to refresh() copies the stdscr to the curscr.
Home (upper left) is (0, 0). Lower right is (LINES-1, COLS-1). Coords are (row, col)

To move the cursor:
move(y, x); // to row y, col x

To clear the screen:
 erase(); // blanks to stdscr
clear(); // erase(), clear term
clrtoeol(); // clear to line end
clrtobot(); // clear to bottom

See the helloworld.c example.
Lecture 8: Terminals
Introduction to Curses

- The integers LINES and COLS contain the number of lines and columns on the screen. Since some terminal screens can be resized it is recommended that you use `getmaxyx()` to get the current screen size.

- There are detailed man pages. See `man ncurses`, `man initscr`, `man move`, `man insch`, etc.
Lecture 8: Terminals
Basic Curses Routines

• To display output:
  
  ```c
  addch(ch);   // display character
  insch(ch);   // insert character
 printw(fmt, var); // formatted
  insertln();  // insert blank line
  beep();      // get user attention
  box(win,vch,hch); // draw a box
  ```

• Many also have mv*() forms that perform cursor movement: `mvaddch(y, x, ch);`
Lecture 8: Terminals
Character Attributes

• Each character can have certain attributes (bold, underline, reverse, blink, etc). Attributes are controlled with the `attron()`, `attroff()`, and `attrset()` routines. See the `attron()` man page for a list of attributes.

• See the `attrib.c` example. (pgs 219-220.)
The input handler is in **canonical** (standard) mode by default. The following routines can be used to control the input mode:

```c
echo();       // turn on char. echo
noecho();     // turn off char. echo
cbreak();     // raw with signals
nocbreak();   // restore std mode
raw();        // raw w/o signals
noraw();      // restore std mode
```
Lecture 8: Terminals

Curses Input

- The following input routines are available:

  keypad(win, flg);  // ESC seq.
  getch();          // get a char
  getstr(str);      // get a C string
  scanw(fmt, varp); // form. input

- In cbreak mode `getch()` returns a character as soon as it is pressed. In standard mode a character is returned only after a newline.

- See `getpass.c` and `keypad.c`. 
The following color routines are available:

```c
has_colors();  // Does term do color?
start_color(); // init. col support
init_pair(...); // set up fg/bg pair
attron(COLOR_PAIR(1));
```

- Variable COLORS contains the # of colors and COLOR_PAIRS the # of color pairs.
- See the color.c example.
Lecture 8: Terminals
Multiple Windows

• The following window routines are available:

  newwin(); // set up new win struct
delwin(); // delete window
touchwin(); // flag window as changed
wrefresh(win); // draw win to screen

• There are also routines of the form
  (waddch, wmove, wprintw, wrefresh)
  that are for working with a specific window.

• See the twowindow.c example.