CS50 ncurses

Overview

ncurses is a C library for writing **text-based user interfaces** that runs portably across various terminals and terminal emulators. ncurses conceives of the terminal screen as a grid of (y, x) character positions, where y counts rows down from the top of the screen and x counts rightwards. It provides an application programing interface (**API**), which is a series of functions for manipulating the terminal screen. Hundreds of terminal-based applications use ncurses.

Key Terms

- ncurses
- text-based user interface
- API
- header file

Using ncurses

To use functions from any library in C, we need to `#include` the **header file** at the top of our source code file. In this case, we'll use `#include <ncurses.h>`. When compiling our code, we'll also have to link the library (with `-lncurses` when working with ncurses, typcially in a file called a Makefile, which tells `make` what to do) so that the resulting object code knows how to execute the functions.

Some Common ncurses Functions

`initscr()` takes no arguments, but, as a side effect, initializes all the appropriate data structures and flushes the screen.
`endwin()` is the antidote to initscr, this time quitting `ncurses` and returning the terminal window to its state preceeding the program.
`getch()` takes no input, but returns a character typed in at runtime by the user.
`move()` moves the cursor to a `(y, x)` position on the screen.
`addch()` adds a character at the cursor's location.
`mvaddch()` takes a `(y, x)` position and a character, combining the above two functions into one.
`mvaddstr()` takes a `(y, x)` position and a string, writing the start of the string at that position and going rightwards.

An example

In the code below, we first initialize ncurses with `initscr()`. Then we use `raw()` to prevent the terminal from buffering the characters that a user may type. That way, the progam can detect as soon as a user types even a single character.

The `for` loop takes care of each line in the diagonal, one by one. Notice how we can use move and addch equivalently to mvaddch. For entire strings, the mvaddstr function can be used.

Notice how we place the characters or strings at `(y, x)` indices, where `y` counts rows down and `x` counts rightward.

This different coordinate system makes more sense for programs that read left to right, top to bottom.

```
1  // initialize ncurses
2  initscr();
3  raw();
4
5  for (int i = 0; i < 10; i++)
6  {
7      move(i, i + 2);
8      addch('*');
9      mvaddstr(i, i + 11, "/////");
10     mvaddch(i, i + 24, '*');
11 }
12 // quit on any input
13 getch();
14
15 // close ncurses
16 endwin();
17 return 0;
```

```
$ make diagonal
$ ./diagonal
```