CS 246E Fall 2020 — ncurses Quick Start Guide

November 18, 2020

1 Compiling

Include the header `<ncurses.h>` and compile with the flag `-lncurses`. Note that if you are not using a Makefile, this flag must come after the files that you are compiling, e.g.

```
g++14 main.cc -lncurses
```

2 Start the ncurses screen

Call the function `initscr()` to begin using ncurses. At the end of the program, you must call `endwin()` to restore the terminal back to normal (if you don't do this, you may find that you are not able to use your terminal even after your program has terminated).

```
#include <ncurses.h>

int main() {
    initscr();
    endwin();
}
```

The above code doesn't do anything (it starts the ncurses screen then immediately closes it). But it is correct. Make sure you can compile it, first and foremost.

3 The default window

In ncurses, a window is a 2-dimensional array of characters that represents all or part of the screen. ncurses supports multiple windows, but this guide will only cover usage of the default window, which has the height and width of the entire screen. The default window is automatically created. It is called `stdscr`. You may not know the height and width of the current screen. The function `getmaxyx(WINDOW *, int y, int x)` gives us this information.

```
#include <ncurses.h>
#include <iostream>

int main() {
    initscr();
    int h, w;
    getmaxyx(stdscr, h, w);
    endwin();
    std::cout << h << "," << w << std::endl;
}
```

You still won't see the ncurses window when running the above code, because it is still open and closed instantly, but now you know the height and width of the current screen.

4 Reading input

Now we want to read input from the window. The function `getch()` reads input from `stdscr`. If we call this in a loop, we can now see the window in action, since the window will stay open while reading characters.

```
#include <ncurses.h>

int main() {
    initscr();
    int h, w;
    getmaxyx(stdscr, h, w);
    int c = 0;
    while (c != 27) { //escape
       c = getch();
    }
    endwin();
}
```

The above code will continually read input until the user presses the escape key. By default, characters that are read are also echoed to the screen, so you will see the characters that you type start filling up the screen from left to right, and the cursor also moves.

You probably don't want this for your Vim implementation, since you want to be able to press keys that do not show up on the screen while using Vim. Calling `noecho()` before doing anything will stop the echoing. Also, calling `keypad(stdscr, TRUE)` will allow you to capture special keys like backspace and delete. Read the manpage for `getch` for more information on this (`man getch`). Why is `c` an `int`? For this very reason. There are typeable keys on the keyboard not in the ASCII table.

```
#include <ncurses.h>

int main() {
    initscr();
    noecho();
    keypad(stdscr, TRUE);
    int h, w;
    getmaxyx(stdscr, h, w);
    int c = 0;
    while (c != 27) { //escape
       c = getch();
    }
    endwin();
}
```

Now the code will behave as before, except you do not see the characters that are being read by ncurses. The screen is simply blank.

5 Output

An ncurses program that does not print characters is a pretty useless program. Let's start printing some characters to the screen. Previously, ncurses automatically printed our input characters from left to right, but we want customized behaviour. I want to print my characters diagonally. The function `addch(chtype ch)` adds a character on the window at the current cursor position. So, if I want to print characters diagonally on the screen, I have to move the cursor diagonally first. The function `move(int y, int x)` moves the cursor to the `y` row and `x` column of the screen. Finally, the function `refresh()` will update the window. You usually have to call this every time you make a change so that the change is actually reflected.

```
#include <ncurses.h>

int main() {
    initscr();
    noecho();
    keypad(stdscr, TRUE);
    int h, w;
    getmaxyx(stdscr, h, w);
    int cursorY = 0;
    int cursorX = 0;
    int c = 0;
    while (c != 27) { //escape
       c = getch();
       addch(c);
       cursorY = (cursorY + 1) % h; // increment cursorY, allow it to wrap around to 0
       cursorX = (cursowX + 1) % w; // same
       move(cursorY, cursorX);
       refresh();
    }
    endwin();
}
```

Now my program takes input from the user then prints it diagonally on the screen.

6 Wrapping up

This guide is intended for you to be able to get started quickly with ncurses, and does not go into depth. Here are more things you may be interested in.

- Multiple windows: The functions described in this guide only deal with `stdscr`, which should be enough to implement a basic version of Vim. But, if you have a reason to require multiple windows at a time, you will have to use extended version of these functions.
- Colouring: I didn't touch on `chtype` in this guide, but this type is not `char`. This type actually allows you to encode display information in the character to display it with a certain color, underline, etc.
- C++: This is a C library. You might find it easier to work with if you make a C++ wrapper to your liking.
- In relation to the above, consider the functions `initsrc()` and `endwin()`. It's annoying to remember to call these at the begin and end of your program, especially if you have a complicated control flow. I wonder if there's a way to guarantee these functions are always called at the right times.
- `ungetch()` is a useful function.