使用 curses 函数库管理基于文本的屏幕

张海宁

贵州大学

hnzhang1@gzu.edu.cn

April 25, 2018

Overview

1. curses
   - use curses
2. curses 库中常用的一些函数
   - 屏幕相关
   - 键盘相关
   - 窗口相关
3. Appendix

curses

curses

curses 是一个适用于unix-like 系统的提供终端控制功能的函数库，其为文本用户界面程序（比如 vi/vim）的开发带来了很大的方便。

use curses

- 引入头文件
  #include<curses.h>
- 编译程序时要使用-lcurses 指明链接库
  gcc curses.c -o window -lcurses

curses.h 一般情况下保存在/usr/include/curses.h

curses ncurses

curses 函数库有多种不同的实现版本，linux 上对应的版本是 ncurses(又称 new curses)。
查看库文件路径下，可以发现 curses 实际上调用的是 ncurses。
ls -al /usr/lib | grep curse
lrwxr-xr-x libcurses.dylib -> libncurses.5.4.dylib
-rwxr-xr-x libncurses.5.4.dylib

stdscr 与 curscr

curses 工作在屏幕、窗口和子窗口之上。无论何时，至少存在一个 curses 窗口，称之为 stdscr，它与物理屏幕的尺寸完全一样。curscr 是指的当前屏幕的样子，直到程序调用 refresh 之后，curses 函数库才会比较 curscr 与 stdscr 之间的不同之处，然后利用这个差异来刷新屏幕。

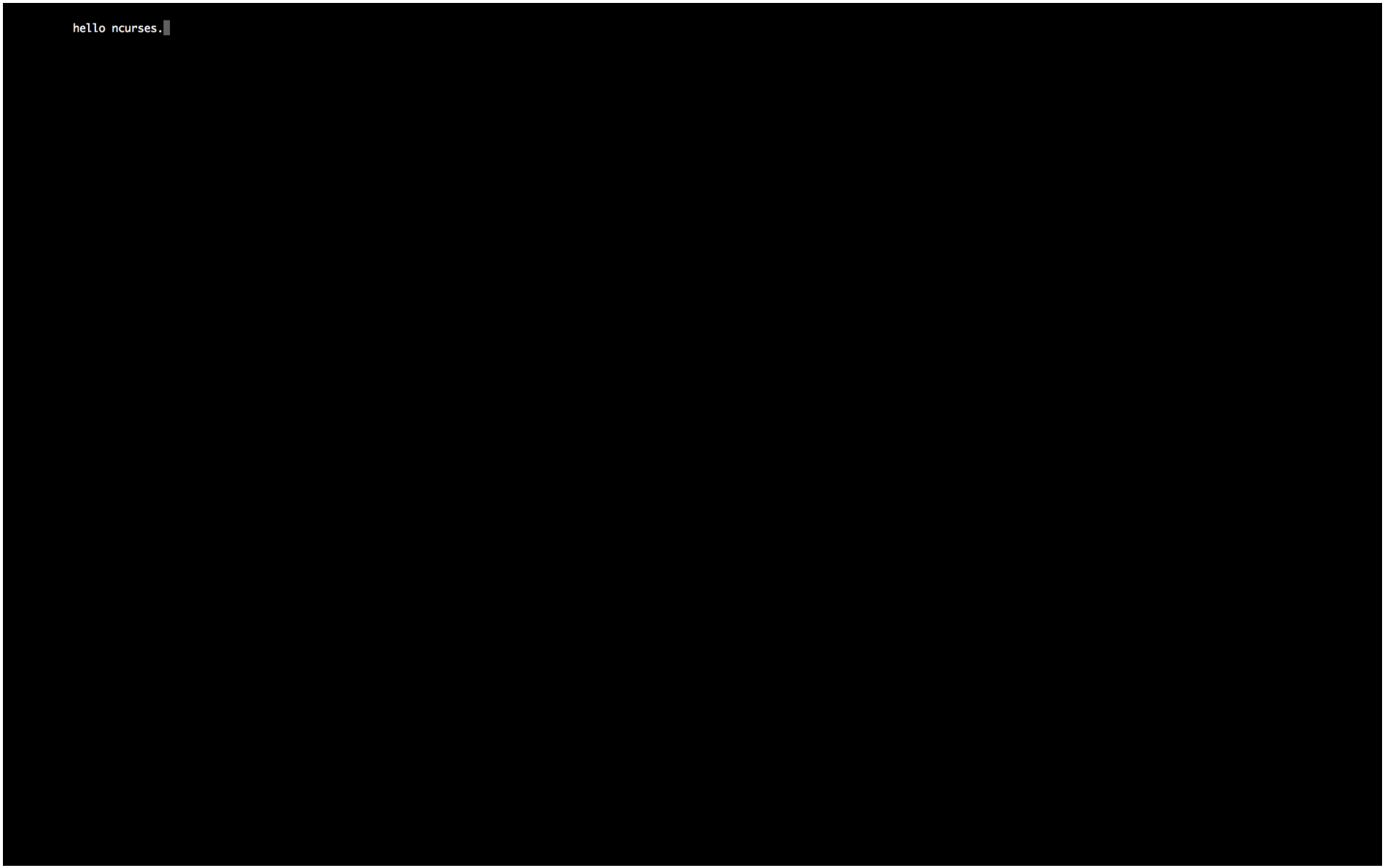
curses demo code

curses.c

```
 cat curses.c
#include<unistd.h>
#include<stdlib.h>
#include<curses.h>

int main(){
   initscr();
   move(1,10);
   printw("%s","hello ncurses.");
   refresh();
   sleep(5);
   endwin();
   exit(0);
}
```

curse demo screenshot



屏幕相关

初始化和重置函数

原型及说明

- WINDOW *initscr(void)
  初始化一个屏幕，必须要调用，一个程序中只能调用一次。返回一个指向结构体 WINDOW 的指针。
- init endwin(void)
  重置终端，使其恢复原来的样子，否则就会一直保持当前的样子，如 ppt 第9页。

输出到屏幕

原型及说明

- int addch(const chtype ch);
  输出一个字符到当前光标位置，光标右移一个位置，会覆盖当前位置原有的字符。
- int addstr(const char *str);
- int refresh(void);
  更新屏幕，将自上次调用 refresh 方法以来所做的改变（比如 addch.addstr 等）显示到屏幕上。
- int beep(void); int flash(void);
  发出声音或闪动屏幕。(这两个方法是会互相调用的，如果终端不支持第一个那就自动调用第二个，如果都不支持就什么也不做。)
- int printw(const char *fmt, ...);
  works like printf, but in a curses environment.

清除屏幕及移动光标

原型及说明

- int clear(void);
  清空屏幕，并且将光标置于屏幕的左上角，需要配合 refresh 来使用。
- int move(int y, int x);
  移动光标到指定的行和列。(注意该方法只会移动逻辑屏幕上的光标位置，下次的输出内容将出现在该位置上；如果想让当前物理屏幕上的光标立刻产生变化，需要在 move 之后就调用 refresh。)

字符属性

每个 curses 字符都可以有特定的属性，该属性控制着字符在屏幕上的显示方式（如果屏幕支持）。预定义属性有：
A_BLINK,A_BOLD,A_DIM,A_REVERSE,A_STANDOUT 和 A_UNDERLINE.

原型及说明

- int attroff(int attrs); int attron(int attrs);
  关闭或启用指定的属性。

键盘相关

键盘工作模式

原型及说明

- - int cbreak(void);
    设置输入模式为字符中止模式。字符一经输入立刻传送给程序。键盘特殊字符也被启用，但一些简单的特殊字符，比如 backspace, 会失去原有功能。
  - int nocbreak(void);
    设置输入模式为行模式。此时键盘特殊字符被启用，按下特定的组合键会产生一个信号。
- int echo(void); int noecho(void);
  开启或关闭字符的回显。(密码。)
- int raw(void); int noraw(void);
  关闭和恢复特殊字符的处理。noraw 还有一个功能是恢复行模式。

读取键盘输入

原型及说明

- int getch(void);
  获取一个字符，并以 int 返回。
- int getstr(char *str);
  获取一个字符序列 (回车提交)，保存到一个字符数组中。
- int scanw(char *fmt, ...);
  类似于 scanf。

窗口相关

窗口

之前的程序所使用的屏幕，都是当前的整个屏幕，从现在开始介绍窗口的使用。

原型及说明

- WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x); 从指定的行 begin_y 和列 begin_x 开始，创建一个 nlines 行和 ncols 列的窗口，返回一个指向新窗口的指针。
- int delwin(WINDOW *win);
  删除一个窗口。
- int box(WINDOW *win, chtype verch, chtype horch);
  围绕一个窗口绘制方框。
- void getyx(WINDOW *win, int y, int x);
  设置指定窗口中当前的光标位置。The getyx macro places the current cursor position of the given window in the two integer variables y and x.

通用函数

之前的一些函数加上一些前缀则会变为通用函数 (适用性更广的函数)。
前缀w用于窗口，前缀mv用于光标移动，前缀mvw用于在窗口之间移动。

原型及说明

- int printw(const char *fmt, ...);
- int wprintw(WINDOW *win, const char *fmt, ...);
- int mvprintw(int y, int x, const char *fmt, ...);
- int mvwprintw(WINDOW *win, int y, int x, const char *fmt, ...);

移动和更新屏幕

原型及说明

- int mvwin(WINDOW *win, int y, int x);
  在屏幕上移动窗口。
- int wrefresh(WINDOW *win);
  refresh 的通用版本，可以指定某个窗口。
- int touchwin(WINDOW *win);
  告诉 curses 函数库某个窗口已发生了更改 (即使其并未更改)，在下次调用 wrefresh 时，会重新绘制该窗口。默认情况下，如果某个窗口的实际内容没有更改的话，调用 wrefresh 不会看到什么效果。但是当调用了 touchwin 再调用 wrefresh 的话，窗口会重新被绘制。
  当屏幕上存在多个窗口时，可通过此函数来安排显示窗口。
- int scroll(WINDOW *win);
  将指定窗口中的内容上卷一行。

子窗口

原型及说明

- WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x);
  创建某个窗口的子窗口。可以使用 mvw 前缀的函数来操纵子窗口。

keypad 模式

对大多数终端来说，解码一些特殊的按键（Home,esc, 方向键等）是一件很困难的事情，因为这些按键往往会发送“以 escape 字符开头的字符串序列”来标识自己。
应用程序需要区分是单独按下了 escape 键，还是按下了某个功能键。
curses 函数库可以通过 keypad 这个函数开启 keypad 模式，来接管特殊按键的识别问题。其在 curses.h 头文件中通过一组以 KEY_ 开头的定义来管理这些特殊按键。

原型及说明

- int keypad(WINDOW *win, bool bf);
  bf 为 true 时开启 keypad 模式。

彩色显示 I

早期只有极少数终端支持彩色显示功能，所以大多数 curses 函数库不支持彩色显示，但是现在的技术发展和人们要求的提高，ncurses 实现了对彩色的支持。
curses 对颜色的支持有个特殊之处：必须同时定义一个字符的前景色和背景色，称之为颜色组合（color pair）。

在使用颜色之前必须先调用以下两个函数：

- bool has_colors(void);
  检查终端是否支持颜色。
- int start_color(void);
  初始化颜色显示功能。

彩色显示 II

初始化了颜色显示以后，可以使用如下的方式来使用色彩方案。

使用色彩方案

- int init_pair(short pair, short f, short b);
  定义一个颜色显示方案 pair，其前景色为 f，背景色为 b。
- int COLOR_PAIR(int pair)
  使用特定组合的颜色方案。
- 将一个窗口后续添加的内容设置为绿色背景和红色前景
  init_pair(1,COLOR_RED, COLOR_GREEN);
  wattron(window_ptr, COLOR_PAIR(1));

pad

在编写高级 curse 程序时，有时候需要先建立一个逻辑屏幕，然后再把它的部分或全部内容输出到物理屏幕上。到目前所学的为止，所有的屏幕都不能大于物理屏幕，为了解决这个问题 curses 提供了一个特殊的数据结构 pad，pad 可以控制尺寸大于物理屏幕的逻辑屏幕。
pad 类似于 WINDOW，所有执行写窗口操作的 curses 函数同样适用于 pad，但 pad 有其特有的创建函数和刷新函数。

pad 特有的一些函数

- WINDOW *newpad(int nlines, int ncols);
  注意其返回的是一个 WINDOW。
- int prefresh(WINDOW *pad, int pminrow, int pmincol, int sminrow, int smincol, int smaxrow, int smaxcol);
  将 pad 上某个位置 (pminrow,pmincol) 开始的区域写到屏幕上指定的显示区域 ((sminrow,smincol)∼(smaxrow,smaxcol))。

作业

1. 设计一个程序用来模拟 terminal 的用户登陆界面。要求用户输入用户名，然后再要求用户输入密码，密码不能显示出来。
2. 尝试编写一个 vim-like 程序，至少可以打开并显示文件内容。

The End

Appendix

参考资料

1. http://heather.cs.ucdavis.edu/~matloff/UnixAndC/CLanguage/Curses.pdf
2. vim source code
   https://www.vim.org/

about man page

The manual is generally split into eight numbered sections, organized as follows (on Research Unix, BSD, macOS and Linux):

section	description
1	General commands
2	System calls
3	Library function(C standard library)
4	Special files(devices) and drivers
5	File formats and conventions
6	Games and screensavers
7	Miscellanea
8	System administration commands and daemons

Table: man page

在终端中运行 man read 与 man 2 read ， 观察其输出的区别。