使用 ncurses 进行颜色编程

作者: Jim Hall 译者: LCTT leemeans | 2018-04-15 09:31

> Jim 给他的终端冒险游戏添加了颜色，演示了如何用 curses 操纵颜色。

在我的使用 ncurses 库进行编程的系列文章的 **第一篇** 和 **第二篇** 中，我已经介绍了一些 curses 函数来在屏幕上作画、从屏幕上查询和从键盘读取字符。为了搞清楚这些函数，我使用 curses 来利用简单字符绘制游戏地图和玩家角色，创建了一个简单的冒险游戏。在这篇紧接着的文章里，我展示了如何为你的 curses 程序添加颜色。

在屏幕上绘图一切都挺好的，但是如果只有黑底白字的文本，你的程序可能看起来很无趣。颜色可以帮助传递更多的信息。举个例子，如果你的程序需要报告执行成功或者执行失败时。在这样的情况下你可以使用绿色或者红色来帮助强调输出。或者，你只是简单地想要“潮艺”一下给你的程序来让它看起来更美观。

在这篇文章中，我用一个简单的例子来展示通过 curses 函数进行颜色操作。在我先前的文章中，我写了一个可以让你在一个粗糙绘制的地图上移动玩家角色的初级冒险类游戏。但是那里面的地图完全是白色和黑色的文本，通过形状来表明是水（ ~ ）或者山（ ^ ）。所以，让我们将游戏更新到使用颜色的版本吧。

颜色要素

在你可以使用颜色之前，你的程序需要知道它是否可以依靠终端正确地显示颜色。在现代操作系统上，此处应该永远为true。但是在经典的计算机上，一些终端是单色的，例如古老的 VT52 和 VT100 终端，一般它们提供黑底白色或者黑底绿色的文本。

可以使用 `has_colors()` 函数查询终端的颜色功能。这个函数将会在终端可以显示颜色的时候返回 `true` ，否则将会返回 `false` 。这个函数一般用于 `if` 语句的头，就像这样：

```
if (has_colors() == FALSE) {
    endwin();
    printf("Your terminal does not support color\n");
    exit(1);
```

在知道终端可以显示颜色之后，你可以使用 `start_color()` 函数来设置 curses 使用颜色。现在是时候定义程序将要使用的颜色了。

在 curses 中，你应该按对定义颜色：一个前景色放在一个背景色上。这样允许 curses 一次性设置两个颜色属性，这也是一般你想要使用的方式。通过 `init_pair()` 函数可以定义一个前景色和背景色并关联到索引数字来设置颜色对。大致语法如下：

```
init_pair(index, foreground, background);
```

控制台支持八种基础的颜色：黑色、红色、绿色、黄色、蓝色、品红色、青色和白色。这些颜色通过下面的名称为你定义好了：

- COLOR_BLACK
- COLOR_RED
- COLOR_GREEN
- COLOR_YELLOW
- COLOR_BLUE
- COLOR_MAGENTA
- COLOR_CYAN
- COLOR_WHITE

应用颜色

在我的冒险游戏中，我想要让草地呈现绿色而玩家的足迹变成不易察觉的绿底黄色点迹。水应该是蓝色，那些表示波浪的 `~` 符号应该是近似青色的。我想让山（`^`）是灰色的，但是我可以用白底黑色文本做一个可用的折中方案。（LCTT 译注：意为终端预设的颜色没有灰色，使用白底黑色文本做一个折中方案）为了让玩家的角色更易见，我想要使用一个刺目的品红底红色设计。我可以像这样定义这些颜色对：

```
start_color();
init_pair(1, COLOR_YELLOW, COLOR_GREEN);
init_pair(2, COLOR_CYAN, COLOR_BLUE);
init_pair(3, COLOR_BLACK, COLOR_WHITE);
init_pair(4, COLOR_RED, COLOR_MAGENTA);
```

为了让颜色对更容易记忆，我的程序中定义了一些符号常量：

```
#define GRASS_PAIR    1
#define EMPTY_PAIR    1
#define WATER_PAIR    2
#define MOUNTAIN_PAIR 3
#define PLAYER_PAIR   4
```

有了这些常量，我的颜色定义就变成了：

```
start_color();
init_pair(GRASS_PAIR, COLOR_YELLOW, COLOR_GREEN);
init_pair(WATER_PAIR, COLOR_CYAN, COLOR_BLUE);
init_pair(MOUNTAIN_PAIR, COLOR_BLACK, COLOR_WHITE);
init_pair(PLAYER_PAIR, COLOR_RED, COLOR_MAGENTA);
```

组合。为了设置颜色，应该在调用像 `mvaddch()` 这样的函数之前使用 `attron()`，然后通过 `attroff()` 关闭颜色属性。例如，在我绘制玩家角色的时候，我应该这样做：

```
attron(COLOR_PAIR(PLAYER_PAIR));
mvaddch(y, x, PLAYER);
attroff(COLOR_PAIR(PLAYER_PAIR));
```

记住将颜色应用到你的程序对你如何查询屏幕有一些微妙的影响。一般来讲，由 `mvinch()` 函数返回的值是没有带颜色属性的类型 `chtype`，这个值基本上是一个整型值，也可以当作整型值来用。但是，由于使用颜色添加了额外的属性到屏幕上的字符上，所以 `chtype` 按照扩展的位模式携带了额外的颜色信息。一旦你使用 `mvinch()`，返回值将会包含这些额外的颜色值。为了只提取文本值，例如在 `is_move_okay()` 函数中，你需要和 `A_CHARTEXT` 做 `&` 位运算：

```
int is_move_okay(int y, int x)
{
    int testch;

    /* return true if the space is okay to move into */

    testch = mvinch(y, x);
    return (((testch & A_CHARTEXT) == GRASS)
            || ((testch & A_CHARTEXT) == EMPTY));
}
```

通过这些修改，我可以用颜色更新这个冒险游戏：

```
/* quest.c */

#include <curses.h>
#include <stdlib.h>

#define GRASS     ' '
#define EMPTY     '.'
#define WATER     '~'
#define MOUNTAIN  '^'
#define PLAYER    '*'

#define GRASS_PAIR     1
#define EMPTY_PAIR     1
#define WATER_PAIR     2
#define MOUNTAIN_PAIR  3
#define PLAYER_PAIR    4

int is_move_okay(int y, int x);
void draw_map(void);

int main(void)
{
    int y, x;
    int ch;

    /* 初始化curses */

    initscr();
    keypad(stdscr, TRUE);
    cbreak();
    noecho();

    /* 初始化颜色 */

    if (has_colors() == FALSE) {
        endwin();
        printf("Your terminal does not support color\n");
        exit(1);
```

```
    start_color();
    init_pair(GRASS_PAIR, COLOR_YELLOW, COLOR_GREEN);
    init_pair(WATER_PAIR, COLOR_CYAN, COLOR_BLUE);
    init_pair(MOUNTAIN_PAIR, COLOR_BLACK, COLOR_WHITE);
    init_pair(PLAYER_PAIR, COLOR_RED, COLOR_MAGENTA);

    clear();

    /* 初始化探索地图 */

    draw_map();

    /* 在左下角创建新角色 */

    y = LINES - 1;
    x = 0;

    do {

        /* 默认情况下，你获得了一个闪烁的光标--用来指明玩家 * */

        attron(COLOR_PAIR(PLAYER_PAIR));
        mvaddch(y, x, PLAYER);
        attroff(COLOR_PAIR(PLAYER_PAIR));
        move(y, x);
        refresh();

        ch = getch();

        /* 测试输入键值并获取方向 */

        switch (ch) {
        case KEY_UP:
        case 'w':
        case 'W':
            if ((y > 0) && is_move_okay(y - 1, x)) {
                attron(COLOR_PAIR(EMPTY_PAIR));
                mvaddch(y, x, EMPTY);
                attroff(COLOR_PAIR(EMPTY_PAIR));
                y = y - 1;
            }
            break;
        case KEY_DOWN:
        case 's':
        case 'S':
            if ((y < LINES - 1) && is_move_okay(y + 1, x)) {
                attron(COLOR_PAIR(EMPTY_PAIR));
                mvaddch(y, x, EMPTY);
                attroff(COLOR_PAIR(EMPTY_PAIR));
                y = y + 1;
            }
            break;
        case KEY_LEFT:
        case 'a':
        case 'A':
            if ((x > 0) && is_move_okay(y, x - 1)) {
                attron(COLOR_PAIR(EMPTY_PAIR));
                mvaddch(y, x, EMPTY);
                attroff(COLOR_PAIR(EMPTY_PAIR));
                x = x - 1;
            }
            break;
        case KEY_RIGHT:
        case 'd':
        case 'D':
            if ((x < COLS - 1) && is_move_okay(y, x + 1)) {
                attron(COLOR_PAIR(EMPTY_PAIR));
                mvaddch(y, x, EMPTY);
                attroff(COLOR_PAIR(EMPTY_PAIR));
                x = x + 1;
            }
            break;
```

```
    while ((ch != 'q') && (ch != 'Q'));

    endwin();

    exit(0);
}

int is_move_okay(int y, int x)
{
    int testch;

    /* 当空白处可以进入的时候返回true */

    testch = mvinch(y, x);
    return (((testch & A_CHARTEXT) == GRASS)
            || ((testch & A_CHARTEXT) == EMPTY));
}

void draw_map(void)
{
    int y, x;

    /* 绘制探索地图 */

    /* 背景 */

    attron(COLOR_PAIR(GRASS_PAIR));
    for (y = 0; y < LINES; y++) {
        mvhline(y, 0, GRASS, COLS);
    }
    attroff(COLOR_PAIR(GRASS_PAIR));

    /* 山峰和山路 */

    attron(COLOR_PAIR(MOUNTAIN_PAIR));
    for (x = COLS / 2; x < COLS * 3 / 4; x++) {
        mvvline(0, x, MOUNTAIN, LINES);
    }
    attroff(COLOR_PAIR(MOUNTAIN_PAIR));

    attron(COLOR_PAIR(GRASS_PAIR));
    mvhline(LINES / 4, 0, GRASS, COLS);
    attroff(COLOR_PAIR(GRASS_PAIR));

    /* 湖 */

    attron(COLOR_PAIR(WATER_PAIR));
    for (y = 1; y < LINES / 2; y++) {
        mvhline(y, 1, WATER, COLS / 3);
    }
    attroff(COLOR_PAIR(WATER_PAIR));
}
```

你可能不能认出所有为了在冒险游戏里面支持颜色需要的修改，除非你目光敏锐。 `diff` 工具展示了所有为了支持颜色而添加的函数或者修改的代码：

```
$ diff quest-color/quest.c quest/quest.c
12,17d11
< #define GRASS_PAIR     1
< #define EMPTY_PAIR     1
< #define WATER_PAIR     2
< #define MOUNTAIN_PAIR  3
< #define PLAYER_PAIR    4
<
33,46d26
<     /* initialize colors */
<
<     if (has_colors() == FALSE) {
<     endwin();
<     printf("Your terminal does not support color\n");
```

```
<
<     start_color();
<     init_pair(GRASS_PAIR, COLOR_YELLOW, COLOR_GREEN);
<     init_pair(WATER_PAIR, COLOR_CYAN, COLOR_BLUE);
<     init_pair(MOUNTAIN_PAIR, COLOR_BLACK, COLOR_WHITE);
<     init_pair(PLAYER_PAIR, COLOR_RED, COLOR_MAGENTA);
<
61d40
<    attron(COLOR_PAIR(PLAYER_PAIR));
63d41
<    attroff(COLOR_PAIR(PLAYER_PAIR));
76d53
<            attron(COLOR_PAIR(EMPTY_PAIR));
78d54
<            attroff(COLOR_PAIR(EMPTY_PAIR));
86d61
<            attron(COLOR_PAIR(EMPTY_PAIR));
88d62
<            attroff(COLOR_PAIR(EMPTY_PAIR));
96d69
<            attron(COLOR_PAIR(EMPTY_PAIR));
98d70
<            attroff(COLOR_PAIR(EMPTY_PAIR));
106d77
<            attron(COLOR_PAIR(EMPTY_PAIR));
108d78
<            attroff(COLOR_PAIR(EMPTY_PAIR));
128,129c98
<     return (((testch & A_CHARTEXT) == GRASS)
<         || ((testch & A_CHARTEXT) == EMPTY));
---
>     return ((testch == GRASS) || (testch == EMPTY));
140d108
<     attron(COLOR_PAIR(GRASS_PAIR));
144d111
<     attroff(COLOR_PAIR(GRASS_PAIR));
148d114
<     attron(COLOR_PAIR(MOUNTAIN_PAIR));
152d117
<     attroff(COLOR_PAIR(MOUNTAIN_PAIR));
154d118
<     attron(COLOR_PAIR(GRASS_PAIR));
156d119
<     attroff(COLOR_PAIR(GRASS_PAIR));
160d122
<     attron(COLOR_PAIR(WATER_PAIR));
164d125
<     attroff(COLOR_PAIR(WATER_PAIR));
```

开始玩吧--现在有颜色了

程序现在有了更舒服的颜色设计了，更匹配原来的桌游地图，有绿色的地、蓝色的湖和壮观的灰色山峰。英雄穿着红色的制服十分夺目。

图 1. 一个简单的带湖和山的桌游地图

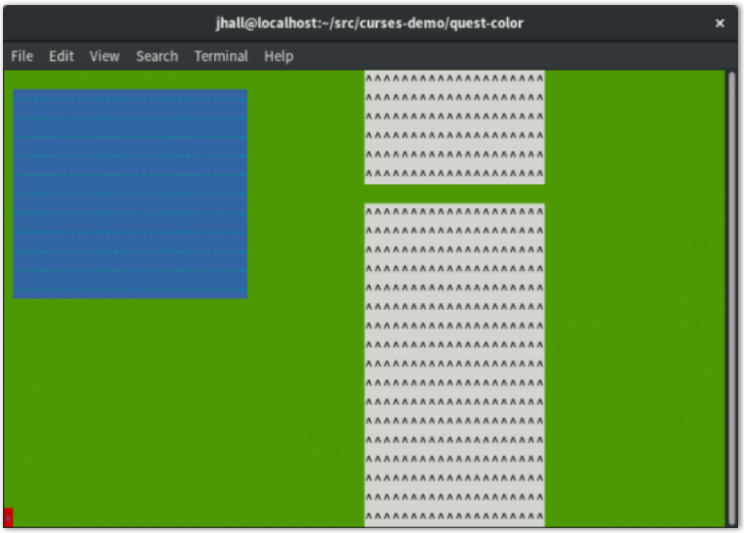

图 2. 玩家站在左下角

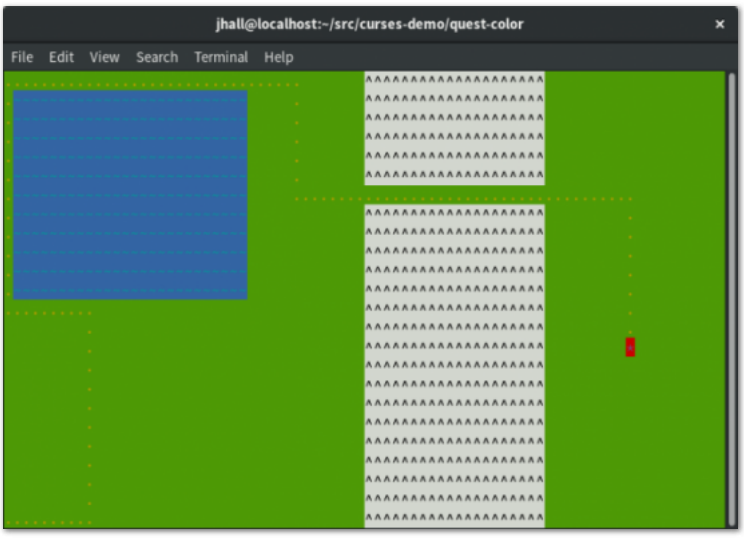

图 3. 玩家可以在游戏区域移动，比如围绕湖，通过山的通道到达未知的区域。

通过颜色，你可以更清楚地展示信息。这个例子使用颜色指出可游戏的区域（绿色）相对着不可通过的区域（蓝色或者灰色）。我希望你可以使用这个示例游戏——你自己的程序的一个起点或者参照。这取决于你需要你的程序做什么，你可以通过 curses 做得更多。

在下一篇文章，我计划展示 ncurses 库的其它特性，比如怎样创建窗口和边框。同时，如果你对于学习 curses 有兴趣，我建议你去读位于 **Linux 文档计划** 的 Pradeep Padala 写的 **NCURSES Programming HOWTO**。

via: http://www.linuxjournal.com/content/programming-color-ncurses

作者： Jim Hall 译者： leemeans 校对： wxy

本文由 LCTT 原创编译， Linux中国 荣誉推出

订阅“Linux 中国”官方小程序来查看

最新评论

发表评论

译自：linuxjournal.com 作者： Jim Hall
原创：LCTT https://linux.cn/article-9546-1.html 译者： leemeans

上一篇：SQL 入门
下一篇：初识 Python： global 关键字

LCTT 译者

leemeans
共计翻译： 7.0 篇 | 共计贡献： 267 天
贡献时间：2018-02-02 -> 2018-10-27
访问我的 LCTT 主页 | 在 GitHub 上关注我

相关阅读

Ncurses

通过 ncurses 在终端创建一个冒险游戏 2018-02-25
ncdu-基于Ncurses的磁盘实用工具 2013-11-19
ncurses 入门指南 2018-02-16
如何在 Linux 中安装 Ncurses 库 2018-05-30