##### **Curses库快速入门**

第一期

2029年9月22日

目录

[Curses库快速入门](#_Toc28160_WPSOffice_Level1) [1](#_Toc28160_WPSOffice_Level1)

[一、初始化](#_Toc718_WPSOffice_Level1) [1](#_Toc718_WPSOffice_Level1)

[1.1使用initscr()对curses初始化](#_Toc718_WPSOffice_Level2) [1](#_Toc718_WPSOffice_Level2)

[1.2 使用endwin()函数中断curses程序](#_Toc24165_WPSOffice_Level2) [2](#_Toc24165_WPSOffice_Level2)

[1.3 使用refresh()和wrefresh()进行屏幕更新](#_Toc30585_WPSOffice_Level2) [2](#_Toc30585_WPSOffice_Level2)

[1.4 CBREAK模式](#_Toc12961_WPSOffice_Level2) [3](#_Toc12961_WPSOffice_Level2)

[1.5 功能键模式keypad()](#_Toc4146_WPSOffice_Level2) [4](#_Toc4146_WPSOffice_Level2)

[二、字符、字符串输出及输入](#_Toc24165_WPSOffice_Level1) [5](#_Toc24165_WPSOffice_Level1)

[2.1 addch()函数](#_Toc9176_WPSOffice_Level2) [5](#_Toc9176_WPSOffice_Level2)

[2.2 addstr()函数](#_Toc6026_WPSOffice_Level2) [5](#_Toc6026_WPSOffice_Level2)

[2.3 printw()函数](#_Toc21877_WPSOffice_Level2) [5](#_Toc21877_WPSOffice_Level2)

[2.4 getch()函数](#_Toc8668_WPSOffice_Level2) [6](#_Toc8668_WPSOffice_Level2)

[2.5 getstr()函数](#_Toc30368_WPSOffice_Level2) [6](#_Toc30368_WPSOffice_Level2)

[2.6 scanw()函数](#_Toc9503_WPSOffice_Level2) [6](#_Toc9503_WPSOffice_Level2)

[三、字符属性](#_Toc30585_WPSOffice_Level1) [7](#_Toc30585_WPSOffice_Level1)

[3.1 curses下支持的字符属性](#_Toc17199_WPSOffice_Level2) [7](#_Toc17199_WPSOffice_Level2)

[3.2 设置和取消字符属性](#_Toc7301_WPSOffice_Level2) [7](#_Toc7301_WPSOffice_Level2)

[四、光标操作](#_Toc12961_WPSOffice_Level1) [9](#_Toc12961_WPSOffice_Level1)

[4.1 移动光标](#_Toc9570_WPSOffice_Level2) [9](#_Toc9570_WPSOffice_Level2)

[4.2 引申：mvfunc(y ， x， [arg，…])](#_Toc32217_WPSOffice_Level2) [10](#_Toc32217_WPSOffice_Level2)

[4.3清除屏幕](#_Toc17979_WPSOffice_Level2) [10](#_Toc17979_WPSOffice_Level2)

[五、颜色属性](#_Toc4146_WPSOffice_Level1) [11](#_Toc4146_WPSOffice_Level1)

[5.1 颜色表定义](#_Toc14067_WPSOffice_Level2) [11](#_Toc14067_WPSOffice_Level2)

[5.2 start\_color()函数](#_Toc2464_WPSOffice_Level2) [12](#_Toc2464_WPSOffice_Level2)

[5.3 使用COLOR\_PAIR(n)属性](#_Toc23495_WPSOffice_Level2) [12](#_Toc23495_WPSOffice_Level2)

[5.4 设置颜色属性](#_Toc7357_WPSOffice_Level2) [13](#_Toc7357_WPSOffice_Level2)

[5.5颜色设置基本模板](#_Toc12748_WPSOffice_Level2) [13](#_Toc12748_WPSOffice_Level2)

##### 

##### **Curses库快速入门**

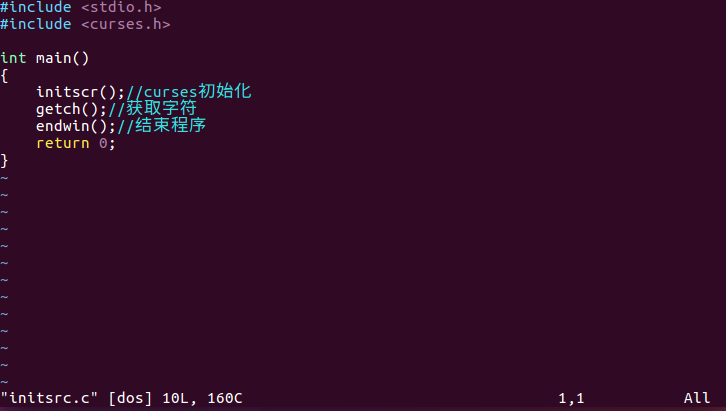
##### **一、初始化**

##### **1.1使用initscr()对curses初始化**

在主函数中设置了信号处理函数之后我们就调用了initscr()，一般情况下在其余的curses函数被调用之前我们就必须首先调用initscr()。initscr()对curses包进行一些初始化的工作，而且在每一个程序里面，这个函数只能调用一次。

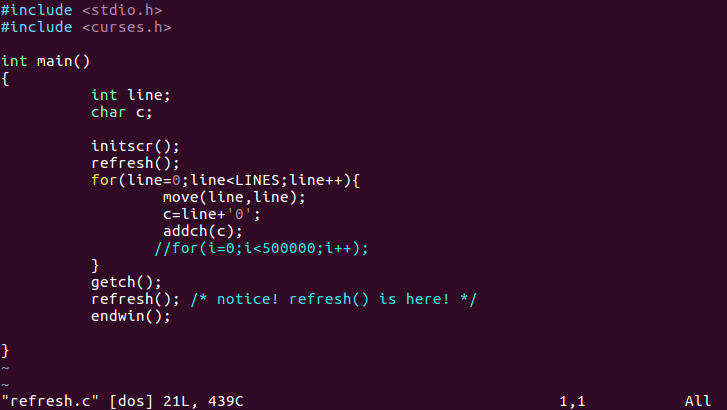
##### **1.2 使用endwin()函数中断curses程序**

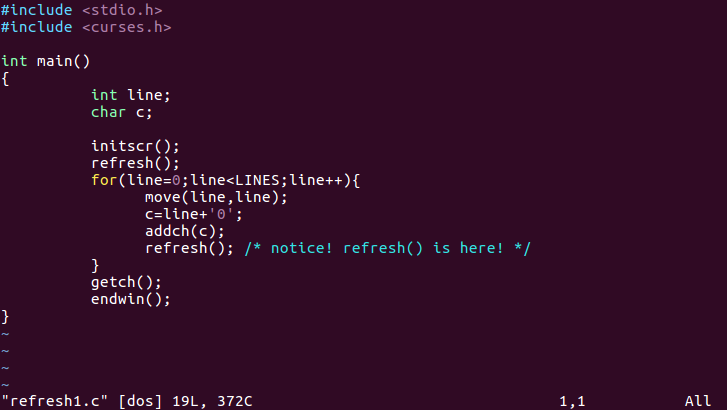
使用curses包的程序必须在程序结束之前能够恢复所有的初始终端设置，因为我们在程序执行过程中可能将终端的某些模式设置或者性能更改掉。这些我们可以通过函数endwin()函数来实现。endwin()是程序最后调用的一个函数，它与initscr()首尾呼应。



##### **1.3 使用refresh()和wrefresh()进行屏幕更新**

为了能够使屏幕更新时候取得最佳效果，即使我们已经对屏幕进行了更改，比如输出了一个字符，curses函数也不会立即更新到终端屏幕上，它将每一次的更新累积起来，只有在程序中调用了函数refresh()或者wrefresh()之后，屏幕才会真正进行更新，从而将终端上的各个窗口之间的相互变化的累积结果反映出来。refresh()更新默认窗口，wrefresh()用来更新程序自定义窗口而不是curses的默认窗口。





从程序的执行速度上可以看出，程序refresh.c的速度远比程序refresh1.c快，从理论上来计算：程序refresh1.c中一共refresh()了LINES次，而程序refresh.c仅更新了一次。如果我们将LINES替换成一个更大的数，那么程序refresh1.c中的速度可想而知。

#### 1.4 CBREAK模式

CBREAK模式也称之为立即输入模式。字符输入在一般的情况下必须加回车键才能完成，这时候退格键是起作用的，输入的字符可以通过BackSpace键删除掉。但是这种操作并不适合交互操作，所以程序有时候把终端模式设置为CBREAK模式。在CBREAK模式下，除了DELETE或者CTRL等仍然被视为特殊控制字符以外，所有的输入字符都被一一立即读取出来。如果没有设置CBREAK模式，从键盘输入的字符都将被存储在缓冲区里面直到输入回车键或者行结束符。**默认情况下CBREAK模式是打开的。**

#### 1.5 功能键模式keypad()

通常情况下功能键比如左移方向键‘←’是不会被读取转换的，因此即使这时候我们调用wgetch()之类的函数也不能将它们读取出来。为了读取这些特殊键，我们必须设置功能键模式。一旦功能键模式开启，键盘上的一些特殊字符比如上下左右以及F1，F2等等按键都可以转换为curses.h内部定义的一些特殊键，这些键通常以“KEY\_”开始，比如KEY\_LEFT、KEY\_DOWN、KEY\_F0等等。**功能键开启函数为keypad(stdscr,ture);**

功能键应用实例：





#### 二、字符、字符串输出及输入

##### **2.1 addch()函数**

**addch()**函数将给定的字符输出到标准屏幕上的当前位置。

例子：**addch(‘a’);**

##### **2.2 addstr()函数**

ddch()函数只能在当前位置增加一个字符，如果需要增加字符串，那就需要使用addstr()函数。

例子：**addstr("line");**

##### **2.3 printw()函数**

printw()函数在屏幕上格式化输出一个或者多个值，这些值包括字符串、字符、十进制数、八进制数、以及二进制数。

例子：**printw("%s %d",name,15);**

##### **2.4 getch()函数**

curses库中的getch()函 数可以从终端键盘读入一个字符，并且返回字符的**整数值**

##### **2.5 getstr()函数**

getstr()函数从终端键盘接受字符串，并且显示在指定的位置上。

例子：

char name[20]；

getstr(name)；

##### **2.6 scanw()函数**

scanw()函数可以格式化输入数据，并把它们拷贝到指定的位置，这个值可以是字符串、字符、十进制、八进制或者二进制数

例子：**scanw("%s %f %c",name,&id,c);**

### 

### 三、字符属性

**3.1 curses下支持的字符属性**

chtype类型包括两部分信息：一部分包含字符本身的信息，另一部分包括与字符相关联的一些属性信息。这些属性允许字符用不同的方式显示，包括反显，加粗，变色，加下划线等等。通过这些方式可以将需要重点突出的文字与其余的文字区分开来。

使用函数 **attrset()** 我们可以更改当前字符的属性。目前curses中支持的一些属性列表如下：

■ A\_NORMAL： //标准的显示模式

■ A\_BLINK： //闪烁属性

■ A\_BOLD： //加粗属性

■ A\_DIM：  //半透明属性

■ A\_REVERSE： //反显属性

■ A\_STANDOUT： //高亮度显示属性

■ A\_UNDERLINE： //加下划线

■ A\_ALTCHARSET： //可代替字符集

■ COLOR\_PAIR(n)： //字符的背景和前景属性

这些属性一般作为参数传递给函数 **attrset()** 以及其余的一些对属性进行操作的函数。当字符作为参数传递给函数的时候我们可以将这些属性直接赋给字符。如果一个字符具有两个或者两个以上的属性，那么这些属性之间使用‘**|**’来对这些属性进行组合。

例如，“A\_UNDERLINE|A\_STANDOUT”属性使得显示字符高亮度显示的同时带有下划线。

#### 3.2 设置和取消字符属性

attron()和attrset()都可以用来设置指定的属性

Attroff()和attrset(0)都可以用来取消指定的属性

例子：

printw(“A word in”)；

attrset(A\_BOLD)；//设置字体加粗

printw(“boldface”)；

attrset(0)；//关闭所有的属性

printw(“really stands out./n”；



运行结果：

### 

### 

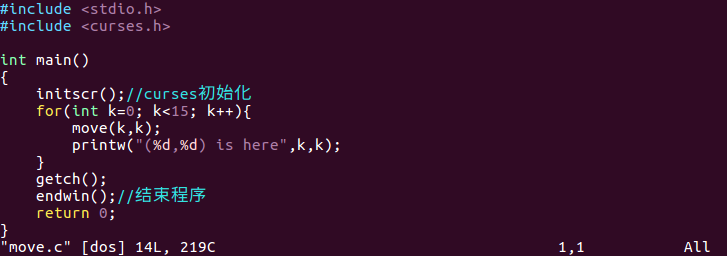
### 四、光标操作

#### 4.1 移动光标

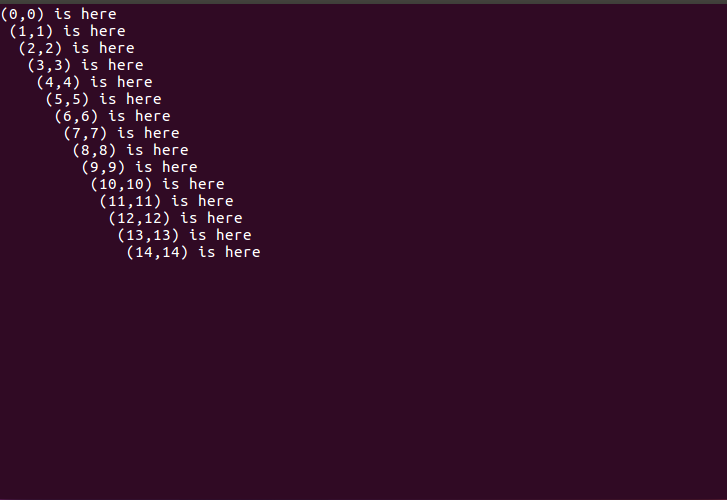
当我们在窗体（包括标准屏幕stdscr）中进行输入输出的时候，逻辑光标会不断的定位于窗体中要进行操作的区域。因此通过移动逻辑光标，你可以在任何时候在窗体的任何部分进行输入输出。urses库中提供的 **move()函数** 可以将逻辑光标移动到给定的位置，这个位置是相对与屏幕的左上角而言，**左上角的坐标为(0，0)**。

**函数用法：move(y，x)；**

参数y是移动后位置的所在的行数，x为新位置所在的的列数。如果移动的目标位置超出了屏幕的范围，则会导致错误。屏幕的行宽和列宽在curses库中定义为(LINES-1，COLS-1)。如果我们需要将光标移动到第五行第四列，则函数代码如下：move(4，3)；



运行结果：



**4.2 引申：mvfunc(y ， x， [arg，…])**

func是原有的普通的函数，只是在func的前面增加mv表示移动位置。函数参数如下：

■ func一般为操作函数的名字，比如addch，addstr等等。

■ y为操作进行时候光标所在的行数，通常也是移动之后的新的光标位置。

■ x为操作进行时候光标所在的列数。

例如：

mvaddch(10 , 5 ,‘X’);

mvprintw(10 , 5 , ”%d”, 15);

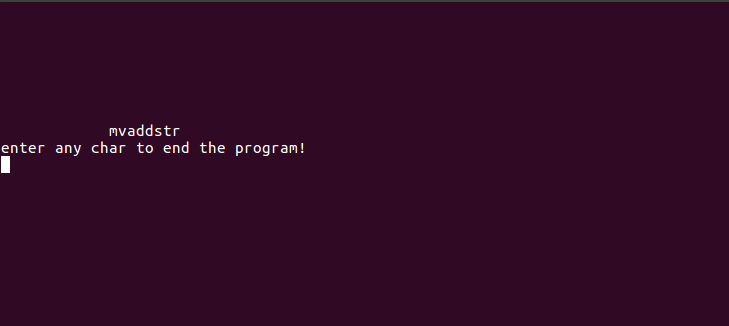
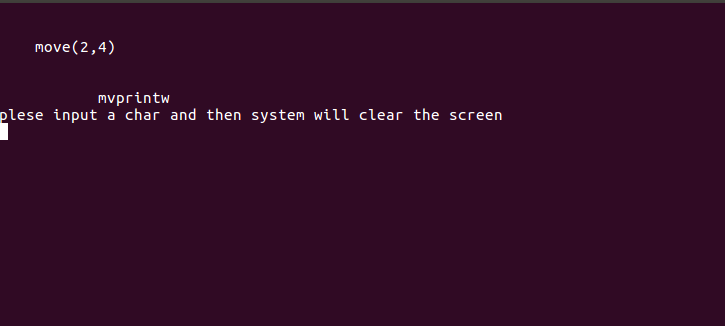
mvaddstr(10 , 5 ,‘abc’);

#### 4.3清除屏幕

#### clear() 和 erase()

### 

运行结果：

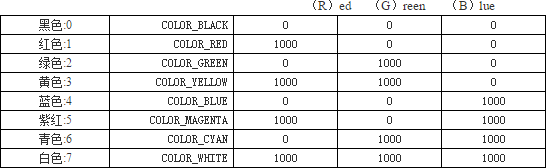


### 五、颜色属性

#### 5.1 颜色表定义

curses库中颜色通常都是配对使用，包括前景色和背景色。为了能够在程序中使用颜色属性，我们首先必须定义使用的颜色，使用它们进行相关初始化工作，并且使用这些颜色创建颜色配对。最后这些颜色配对作为颜色属性供使用。

当curses库 初始化的时候系统将自动的创建一张颜色表产生默认颜色定义。这个颜色定义表中有很多的条目，条目的数量和当前终端一次能够显示的颜色的数量是相匹配的。如下图：



在curses.h中我们定义了下面的一些颜色有关的宏，这些值与默认颜色表相应颜色的位置对应。

COLOR\_BLACK 0

COLOR\_RED 1

COLOR\_GREEN 2

COLOR\_YELLOW        3

COLOR\_BLUE 4

COLOR\_MAGENTA 5

COLOR\_CYAN 　    　  6

COLOR\_WHITE 7

#### 5.2 start\_color()函数

调用start\_color()函数，这样，颜色配对表就会使用函数init\_pair(pair，f，b)进行初始化工作。

注意start\_color()函数应该在输出字符前就调用。最好在initscr()后就开始调用。

#### 5.3 使用COLOR\_PAIR(n)属性

COLOR\_PAIR(n)属性指定当前的前景色和背景色分别为颜色配对表中索引为n的条目中的对应值。例子如下：

init\_pair(1,COLOR\_BLUE,COLOR\_YELLOW);

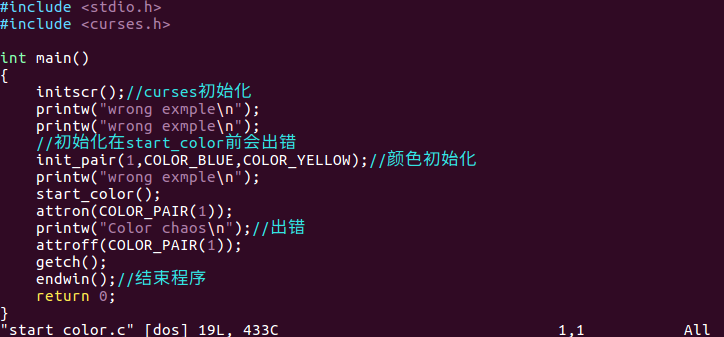
其中1 是color\_pair的代号，COLOR\_BLUE是前景色，COLOR\_YELLOW是背景色。

init\_pair(2,COLOR\_CYAN,COLOR\_MAGENTA);

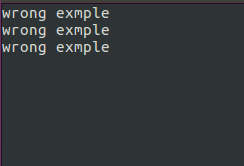
或者

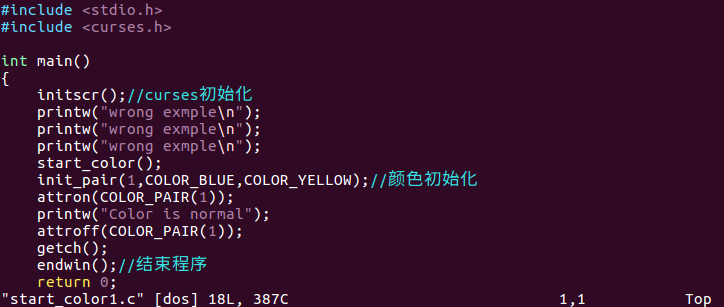
init\_pair(1，4，3)；

init\_pair(2，6，5)；

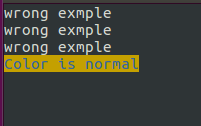


运行结果：





运行结果:



**颜色初始化应该在start\_color()之后。**

#### 5.4 设置颜色属性

Attron(COLOR\_PAIR(1)); 开启颜色对1

Attroff(COLOR\_PAIR(1)); 关闭颜色对1

Attrset(COLOR\_PAIR(1)); 开启颜色对1

Attrset(0) 关闭所有属性（包括字体属性、颜色属性）

**5.5颜色设置基本模板**

// start\_color();

// 开启彩显模式

// init\_pair(1, COLOR\_RED, COLOR\_WHITE);

// 初始化颜色对, 前景色为红, 背景色为白

// attron(COLOR\_PAIR(1));

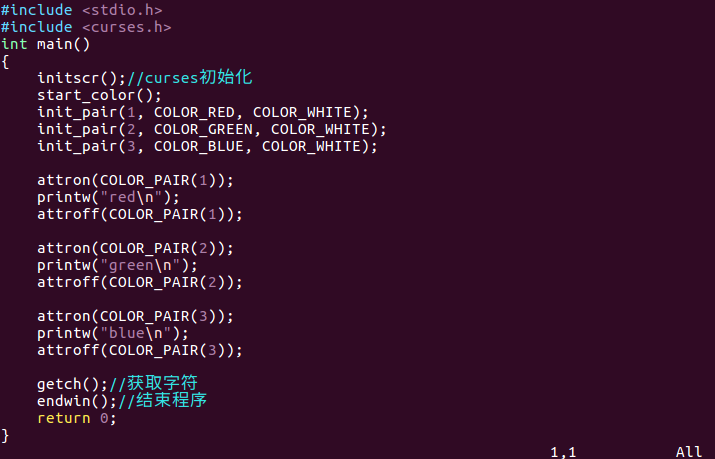
// 开启其中一个颜色对

// ...

// 中间是你要打印的字符

// attroff(COLOR\_PAIR(1));

// 关闭该颜色对



运行结果:

