Curses 库快速入门

第一期

2029 年 9 月 22 日

目录

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。

```
#include <stdio.h>
#include <curses.h>

int main()
{
    initscr();//curses初始化
    getch();//获取字符
    endwin();//结束程序
    return 0;
}
~
~
~
~
~
~
~
~
~
~
~
~
~
"initsrc.c" [dos] 10L, 160C                    1,1           All
```

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。

```
#include <stdio.h>
#include <curses.h>

int main()
{
          int line;
          char c;

          initscr();
          refresh();
          for(line=0;line<LINES;line++){
                  move(line,line);
                  c=line+'0';
                  addch(c);
                 //for(i=0;i<500000;i++);
          }
          getch();
          refresh(); /* notice! refresh() is here! */
          endwin();

}
~
~
"refresh.c" [dos] 21L, 439C                                   1,1           All
```

```
#include <stdio.h>
#include <curses.h>

int main()
{
          int line;
          char c;

          initscr();
          refresh();
          for(line=0;line<LINES;line++){
                move(line,line);
                c=line+'0';
                addch(c);
                refresh(); /* notice! refresh() is here! */
          }
          getch();
          endwin();
}
~
~
~
~
"refresh1.c" [dos] 19L, 372C                                  1,1           All
```

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：

```
#include <stdio.h>
#include <curses.h>
int main(){
    initscr();//初始化屏幕
    clear();//清屏函数
    keypad(stdscr,1);//开启功能键
    int year,month,day;
    printw("please input year , month and day\n");
    scanw("%d %d %d",&year,&month,&day);
    bool button = 1;
    int ch;
    while(button){
        ch = getch();
        switch(ch){
            case KEY_RIGHT: month += 1;printw("%d\n", month); break;
            case KEY_LEFT: month -= 1;printw("%d\n", month); break;
            case KEY_UP: year -= 1;printw("%d\n",year);break;
            case KEY_DOWN: year += 1;printw("%d\n",year); break;
            case 27 : button=0;break;//27表示的是Esc键
        }
    }
    endwin();//关闭curses状态
    return 0;
}
"keyboard.c" 24L, 662C                                1,1           All
```

功能键定义	控制码	描述
KEY_MIN	0401	Curses中定义的最小的键值
KEY_BREAK	0401	Break按键
KEY_DOWN	0402	↓
KEY_UP	0403	↑
KEY_LEFT	0404	←
KEY_RIGHT	0405	→
KEY_HOME	0406	Home键
KEY_BACKSPACE	0407	退格键backspace
KEY_F0	0410	功能键F0
KEY_F(n)	KEY_F0+n	功能键Fn
KEY_DL	0510	行删除键
KEY_IL	0511	行插入键
KEY_DC	0512	字符删除键
KEY_IC	0513	字符插入键
KEY_NPAGE	0522	下一页
KEY_PPAGE	0523	上一页
KEY_END	0550	end按键
KEY_MAX	0777	最大的curses键值

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);

```c
#include <stdio.h>
#include <curses.h>

int main()
{
    initscr();//curses初始化
    addch('c');//输出字符
    addstr(" str");//输出字符串
    printw("%s %d\n","prntw",15);//格式化输出

    printw("please input a char \n");
    int ch = getch();//获取字符
    printw("\nthis char is %c\n",ch);

    printw("please input a str, a float and a char : \n");
    char a,c;
    float b;
    scanw("%s %f %c",&a,&b,&c);//格式化输入

    char str[3];
    printw("please input a str\n");
    getstr(str);//getstr () 从键盘接受字符串并且把它存放在变量name
    printw("this str is %s\n",str);

    printw("Enter any character to end the program !\n");
    getch();//在没输入字符的情况下可以达到暂停程序的作用
    endwin();//结束程序
    return 0;
}
"input_output.c" [dos] 29L, 786C                              1,1           All
```

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

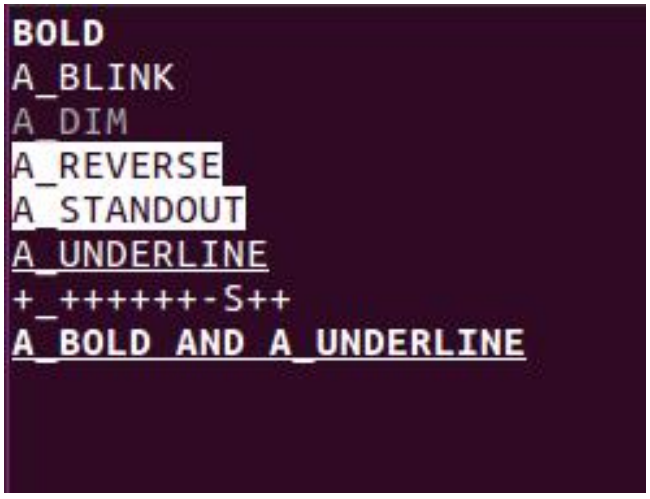
printw(“really stands out./n”;

```c
#include <stdio.h>
#include <curses.h>
int main()
{
    initscr();//curses初始化

    attrset(A_BOLD);//加粗属性
    printw("BOLD\n");
    attrset(A_BLINK);//闪烁属性
    printw("A_BLINK\n");
    attrset(A_DIM);//半透明属性
    printw("A_DIM\n");
    attrset(A_REVERSE);//反显属性
    printw("A_REVERSE\n");
    attrset(A_STANDOUT);//高亮度显示属性
    printw("A_STANDOUT\n");
    attrset(A_UNDERLINE);//加下划线
    printw("A_UNDERLINE\n");
    attrset(A_ALTCHARSET);//可代替字符集
    printw("A_ALTCHARSET\n");
    attrset(A_BOLD | A_UNDERLINE);//加粗并且加下滑线
    printw("A_BOLD AND A_UNDERLINE\n");
    attrset(0);//取消字符属性

    getch();
    endwin();//结束程序
    return 0;
}
"font.c" [dos] 28L, 712C                                    1,1           All
```

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)；

```
#include <stdio.h>
#include <curses.h>

int main()
{
    initscr();//curses初始化
    for(int k=0; k<15; k++){
        move(k,k);
        printw("(%d,%d) is here",k,k);
    }
    getch();
    endwin();//结束程序
    return 0;
}
"move.c" [dos] 14L, 219C                                    1,1           All
```

运行结果：

```
(0,0) is here
 (1,1) is here
  (2,2) is here
   (3,3) is here
    (4,4) is here
     (5,5) is here
      (6,6) is here
       (7,7) is here
        (8,8) is here
         (9,9) is here
          (10,10) is here
           (11,11) is here
            (12,12) is here
             (13,13) is here
              (14,14) is here
```

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()

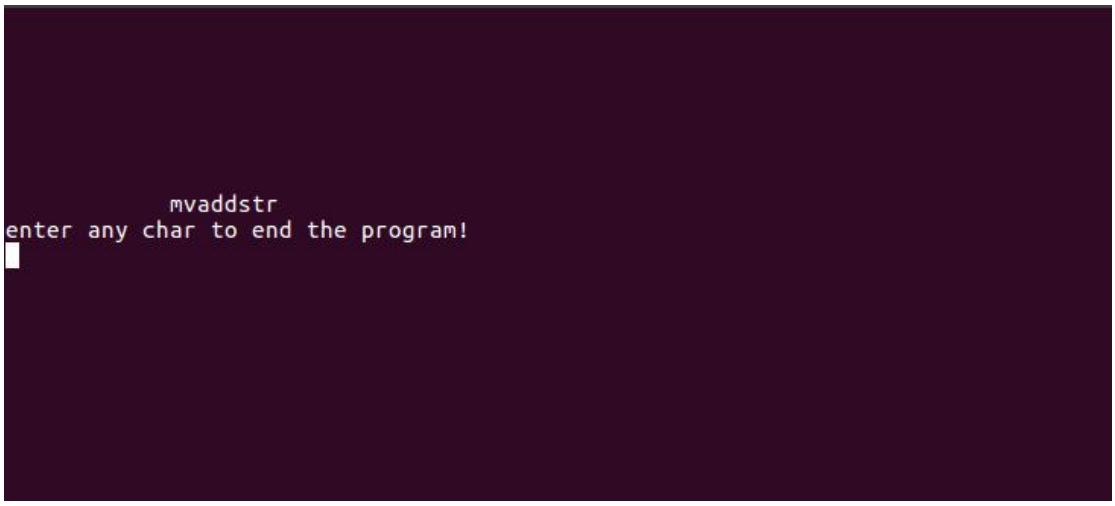
```
#include <stdio.h>
#include <curses.h>

int main()
{
    initscr();//curses初始化
    move(2,4);//移动光标
    printw("move(2,4)");
    mvprintw(5,11,"mvprintw\n");//mvfunc应用
    printw("plese input a char and then system will clear the screen \n");
    getch();
    clear();//清屏
    mvaddstr(7,12,"mvaddstr");//mvfunc
    printw("\nenter any char to end the program!\n");
    getch();
    endwin();//结束程序
    return 0;
}
"guangbiao.c" [dos] 18L, 427C                                  1,1           All
```

运行结果：

```
    move(2,4)


          mvprintw
plese input a char and then system will clear the screen
```



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：

		(R) ed	(G) reen	(B) lue
黑色:0	COLOR_BLACK	0	0	0
红色:1	COLOR_RED	1000	0	0
绿色:2	COLOR_GREEN	0	1000	0
黄色:3	COLOR_YELLOW	1000	1000	0
蓝色:4	COLOR_BLUE	0	0	1000
紫红:5	COLOR_MAGENTA	1000	0	1000
青色:6	COLOR_CYAN	0	1000	1000
白色:7	COLOR_WHITE	1000	1000	1000

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

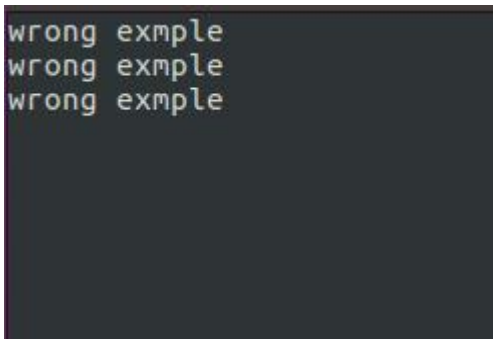
init_pair(1, 4, 3);

init_pair(2, 6, 5);

```
#include <stdio.h>
#include <curses.h>

int main()
{
    initscr();//curses初始化
    printw("wrong exmple\n");
    printw("wrong exmple\n");
    //初始化在start_color前会出错
    init_pair(1,COLOR_BLUE,COLOR_YELLOW);//颜色初始化
    printw("wrong exmple\n");
    start_color();
    attron(COLOR_PAIR(1));
    printw("Color chaos\n");//出错
    attroff(COLOR_PAIR(1));
    getch();
    endwin();//结束程序
    return 0;
}
"start color.c" [dos] 19L, 433C                              1,1           All
```

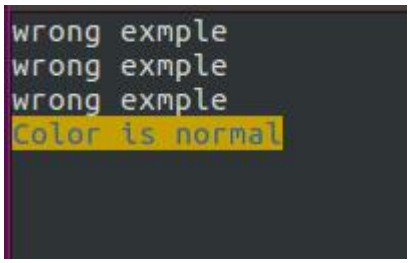
运行结果：




```
#include <stdio.h>
#include <curses.h>

int main()
{
    initscr();//curses初始化
    printw("wrong exmple\n");
    printw("wrong exmple\n");
    printw("wrong exmple\n");
    start_color();
    init_pair(1,COLOR_BLUE,COLOR_YELLOW);//颜色初始化
    attron(COLOR_PAIR(1));
    printw("Color is normal");
    attroff(COLOR_PAIR(1));
    getch();
    endwin();//结束程序
    return 0;
"start_color1.c" [dos] 18L, 387C                    1,1           Top
```

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

```c
#include <stdio.h>
#include <curses.h>
int main()
{
    initscr();//curses初始化
    start_color();
    init_pair(1, COLOR_RED, COLOR_WHITE);
    init_pair(2, COLOR_GREEN, COLOR_WHITE);
    init_pair(3, COLOR_BLUE, COLOR_WHITE);

    attron(COLOR_PAIR(1));
    printw("red\n");
    attroff(COLOR_PAIR(1));

    attron(COLOR_PAIR(2));
    printw("green\n");
    attroff(COLOR_PAIR(2));

    attron(COLOR_PAIR(3));
    printw("blue\n");
    attroff(COLOR_PAIR(3));

    getch();//获取字符
    endwin();//结束程序
    return 0;
}
                                                    1,1           All
```

运行结果：