C语言 curses 函数库与屏幕编辑

曹福元　（南京大学技术部）

摘要：本文对使用 C 语言设计多窗口编辑，以实现 *Foxbase* 的@*get—read* 功能为例，介绍 *curses* 函数的用法，并给出了源程序。

C 语言具有程序简洁、灵活性强、移植性能好、目标代码质量高等特性，尤其是它丰富的函数调用，为设计高质量的、人机界面友好的程序提供了理想工具。因而受到程序设计人员的青睐。特别是近几年来，随着关系数据库管理系统 Informix、Unify、Oracle 在我国普及使用，C 语言作为其宿主语言得到了越来越广泛的使用。

但是，对程序设计人员来说，在使用 C 语言开发和编制程序时，遇到的一个棘手的问题就是屏幕的编辑问题。在 Foxbase 管理系统中，系统本身提供了格式输入语句。如@getread 语句，它为程序设计人员提供了一个简便、清晰、便于组织形式多样的输入数据屏幕的手段。可以很方便的实现多个窗口的编辑，即多个窗口间的上下移动、字符的增加、插入、删除等功能。可以说，任何一个信息管理系统都离不开数据的输入和修改，而数据的输入、修改都离不开屏幕的设计。C 语言标准函数库并未提供类似于 Foxbase 中的@gettead 函数。但是 C 语言 curses 函数库提供了一组基本的屏幕处理函数或称为窗口编辑函数。程序设计人员可通过调用 curses 函数，并且根据具体的要求来设计屏幕编辑程序。本文拟就使用 curses 库所提供的窗口编辑函数，实现 Foxbase 的@get—read 功能为例，讨论窗口函数的使用方法。并给出了部分程序示例，供同行们参考。

屏幕处理函数通过内存里的中间“屏幕”和“窗口”存取终端屏幕。一个屏幕代表整个终端屏幕的样子，一个窗口代表终端屏幕的某一部分的样子，它共享另一个窗口（通常为标准终端屏幕）的所有或部分的字符空间，并且提供了一个交互方法来存取指定空间的字符。定义窗口函数调用形式如下：

swin＝subwin(win, lines, cols, begin—y, begin—x)

其中，swin 为接收回值的指针；win 为指向窗口的指针，一般为 stdcer 即标准屏幕指针，该窗口含有新的子窗口；lines 和 cols 是整数值，分别给出定义的子窗口的总的行数、列数；begin—y 和 begin—x 则定义新辟子窗口在标准屏幕位置上的左上角的行、列下标。

例：WINDOW　* wp;

wp＝subwin(stdscr, 10, 10, 5, 5);

则定义了一个子窗口，位于标准终端屏幕的第 5 行、第 5 列，窗口大小为 10 行 10 列。

窗口 wp 为 WINDOW 的结构指针。WINDOW 窗口结构定义如下：

```
struct  win st{
        short       cury, curx;
        short       maxy, maxx;
        short       begy, begx;
        short       flags;
        bool        clear;
        bool        leave;
        bool        scroll;
#ifdef sysvr2
        bool        keypad;
#ifdef
#ifdef  INTLCHR
        char        * * so;
        char        * sobase;
#endif
        char        * * y;
        char        * ybase;
        short       * firstch;
        short       * lastch;
```

```
    struct win st *  nextp, *  orig;
};
#define WINDOW struct  win st
```

其中：

-cury-curx--光标位于当前窗口的当前行、列座标。

-maxy-maxx--当前窗口的最大行数、最大列数。

-begy-begx--当前窗口位于标准屏幕上的起始行、起始列座标。

需要说明的是，在 WINDOW 窗口结构中，其中-y 被定义为指向字符型指针的指针。它用以存放当前窗口数据。换句话说，它是当前窗口数据的映射区。一般的 C 语言手册中并未介绍。在实际编程过程中，它是十分有用的。在编制屏幕编辑程序时，可采用 curses 窗口函数库所提供的基本函数实现光标的移动、字符的增、插、删、改及屏幕显示等功能，而借助于-y 获取编辑完毕的数据。这种方法为用户编制全屏幕编辑函数提供了一个简便的手段。

curses 函数库提供了一组字符的读、取、增、插、删、改、字符显示及窗口操作等函数(详见有关技术资料)。

如何使用这些函数来实现多个窗口的编辑功能，即类似于 Foxbase 中的@get-read 语句功能。在讨论之前，有必要对 curses 的输入输出模式作一说明：

raw()-原始模式。它使得在键盘上输入的每个字符，作为直接的输入来发送，取消编辑键和信号键的功能，并且取消换行符到换行符和回车符的映射。

echo()-为终端设置 ECHO 模式，使得在键盘上输入的每个字符显示在终端屏幕上。

noecho()-清除终端的 ECHO 方式，这种方式禁止从键盘输入的字符显示在终端上，即关闭回应模式。

noraw()-清除终端的原始模式。恢复终端正常的编辑和信号产生功能。

多窗口编辑程序的说明；

功能键的用法：

左、右光标：用于当前窗口的光标移动。

光标上箭头：退出当前窗口，返回到上一个窗口。

光标下箭头：退出当前窗口，进入下一个窗口。

回车键：退出当前窗口，进入下一个窗口。

x：非编辑状态下，删除光标当前字符。

i：非编辑状态下，进入插入状态。

e：进入编辑状态。

程序示例见下。

本文对单行多窗口编辑程序的编程思想、步骤进行了讨论。因篇幅有限，该程序未涉及汉字的处理。在实际编程中，若涉及到汉字的编辑，就必须考虑汉字的特殊情况。诸 如字符插入时，窗口尾 部可能出现的半个汉字的问题；字符删除时，对汉字的删除特殊处理等。

参考文献：

[1][美]Herbert Schildt C: The Complete Reference

[2]金茂忠 主编，C 程序设计高级教程。

[3]中科院软件研究所编译，XENIX 开发系统 C 语言参考手册与库指南。

[4]陈礼民 江礼璋 编，C 程序设计语言及其应用。

[5]孙玉方 张乃孝编，实用 C 语言程序设计教程。

```
/ * * * * * * * * * * * * * * * * * * * * * * * * *
C 语言编程实现 Foxbase 的@get read 功能程序示例
* * * * * * * * * * * * * * * * * * * * * * * * * /
#define MAXX (sizeof(arr) / sizeof(SOR ARR))
#include "stdio.h"
#include "ctype.h"
#include "malloc.h"
#include "tcap.h"
typedef struct {
        int begy;       / * 起始行 * /
        int begx;       / * 起始列 * /
        int len;        / * 接收字串长度 * /
        char * buf;     / * 字符串指针 * /
        WINDOW * WP;    / * 窗口指针 * /

    }SOR  ARR;
SOR  ARR arr[]={
          {2,9,10}
          {2,27,8}
          {2,43,10}
          {2,61,8}
          {2,72,6}
          {3,9,10}
```

```
        {3,27,8}
        {3,43,10}
        {3,61,8}
        {3,72,6}
};
main()
{
int i,j;
initscr();      /*初始化屏幕处理函数*/
raw();          /*为终端设置原始模式*/
noecho();       /*清除终端 ECHO 方式*/
init buf();     /*字串指针初始化*/
for(i=0;i<MAXX;i++){    /*定义子窗口*/
  arr[i].wp=subwin(stdscr,1,arr[i].len,arr[i].begy,arr[i].begx);
  werase(arr[i].wp);
  wrefresh(arr[i].wp);
}
Get read();     /*调用多窗口编辑函数*/
Free buf();     /*释放分配的存储空间*/
endwin();       /*恢复终端状态结束屏幕处理*/
exit(0);
}
/**********
函数 Get read()
**********/
int Get read()
{
int i,j,cc;
for(i=0;i<MAXX;I++){
  mvwaddstr(arr[i].wp,0,0,arr[i].buf);
  wrefresh(arr[i].wp);
}
for(j=0;j<MAXX;j++){
  cc=scr edit(arr[i].wp);    /*窗口编辑*/
  for(j=0;j<arr[i].wp->maxx;j++){/*通过 WP->-Y
获取窗口数据,送 buf 中*/
    arr[i].buf[j]=arr[i].wp-> y[0][j];
  }
  arr[i].but[j]='\0';
  if(cc=-1){
    if(i,=0) i=-1;
    else
      i-=2;
  }
}
return(0);
}
/*******
函数 Scr edit()
********/
int scr edit(wp)
window *wp;
{
int i,c;
int ins,esc,edit,our flag;
ins=0;
esc=0;
edit=1;
our fiag=0;
wmove(wp,0,0);
wrefresh(wp);
while(c=wgetch(wp))!='\n'&&c!='\r'){
  if(c='\c33'){
    esc=1;
    edit=0;
    ins=0;
    continue;
  }
  if(c='['&&esc=1){
    our flag=1;
    continue;
  }
  if(our flag){
    switch(c){
     case 'A';        /*光标上箭头*/
        return(-1);
     case 'B';        /*光标下箭头*/
        return(1);
     case'c';         /*光标右箭头*/
        if(c='C'&&wp->-arx=wd->-maxx-1)return(1);
        else{
          wmove(wp,wp->-ary,++wp->-curx);
          wrefresh(wp);
        }
              bread;
     case'd';      /*光标左箭头*/
```

```
            if (wp-)-curx=0c='d')retrurn(-1);
            else{
              wmove(wp,wp-)->-cury,-wp->-curx);
              wrefresh(wp);
            }
              break;
            }
            cur-flag=0;
            continue;
}
if(c='i'&&!edit){
  ins=1
  edit=1;
  cur-flag=0;
  esc=0;
  continue;
}
  if(c='x'&&!edit){
    wdelch(wp);
    wmove(wp,wp->-cury,-wp->curx);
    wrefresh(wp);
    continue;
}
if(c='e'&&!edit){
  edit=1;
  cur-flag=0;
  esc=0;
  continue;
    }
if(ins){
  winsch(wp,c);
  if(wp->-curx<wp->-maxx)
    wmove(wp,0++wp->-curx);
    wrefresh(wp);
    continue;
}
if(isprint(c)!edit)
  continue;
else{
      waddch(wp,c);
        wrefresh(wp);
      }
}
return(0);
}
/*******
函数 init-buf()
*********/
int init-buf()
{
itn k;
char *p,*malloc();
for (k=0;k<MAXX;k++){
  p=(char *)malloc(arr[k].len+1);
  if (!p)return(-1);
  else
    arr[k].buf=p;
}
return(0)
}
/******
函数 Free-buf()
*******/
int Free-buf()
{
int k;
  for(k=0;k<MAXX;k++){
    free(arr[K].buf);
  }
return(0);
}
```

※ ※ ※ ※ ※ ※ ※ ※ ※ ※ ※ ※ ※ ※ ※ ※ ※

投稿须知

1.内容开门见山,文笔简炼通顺

2.图形正规

3.程序一律上机通过并打印清楚

※ ※ ※ ※ ※ ※ ※ ※ ※ ※ ※ ※ ※ ※ ※ ※ ※