UNIX下c语言的图形编程--curses.h 函式库

18 unknown 秦皇天下・学院 http://www.qhtx.net/edu

相信您在网路上一定用过如 tin,elm 等工具，这些软体有项共同的特色，即他们能利用上下左右等方向键来控制游标这些程式的画面也较为美观. 对 Programming 有兴趣的朋友一定对此感到好奇，也许他能在 PC 上用 Turbo C 轻易地写出但当他将相同的程式一字不变地移到工作站上来编译时，却出现一堆抓也抓不完的错误. 其实，原因很简单，他使用的函式有定义的. 有些在 Turbo-C 上被广泛使用的一些函式，可能在 UNIX 上是不被定义的.

为了因应网路上各式各样的终端机形态 (terminal)，UNIX 上特别发展出一套函式库，专门用来处理 UNIX 上游标移动及萤篇文章要为您介绍的 - curses.h 函式库. 利用这个函式库，您也可以写出像 elm 般利用方向键来移动光棒位置的程式. (上选课程式，及程式服务界面，即是笔者利用 curses 发展而成的)

■ curses 的历史与版本

cureses 最早是由柏克莱大学的 Bill Joy 及 Ken Arnold 所发展出来的. 当时发展此一函式库主要原因是为了提高程式对而设计的. 因此，利用 curses 发展出来的程式将和您所使用的终端机无关.也就是说，您不必担心您的程式因为换了一部终对程式设计师而言，尤其是网路上程式的撰写，是件相当重要的一件事.
curses之所以能对上百种以上的终端机工作，是因为它将所有终端机的资料，存放在一个叫 termcap 的资料库，(而在第二中，新版的 curses 以 terminfo 取代原来的 termcap). 有了这些记录，程式就能够知道遇到哪一种终端机时，须送什麼字置,送什麼字元才能清除整个萤幕清除. (* 注一)

另外，本文的介绍以 System V的curses 版本为主.

■ 如何在您的程式使用 curses ?

在您的C程式的档头将 include 进来.当您引进curses.h 这个函式库後，系统会自动将和一并 include 进来.另外，在Syst式库也将一并include进来.

```
#include

main()
{
: :
: :
}
```

当然, 您的系统内必须放有 curses.h 这个函式库.

■ 如何编译(compile)

当您编辑好您的程式, 在 UNIX 提示符号下键入:

```
% /usr/5bin/cc [file.c] -lcurses
^^^^^^^
```

引进 curses.h 这个 library

或 % /usr/5bin/cc [file.c] -lcurses -ltermlib

(*注二)

■ 如何开始我的第一个 curses 程式?

在开始使用 curses 的一切命令之前, 您必须先利用 initscr()这个函式
来开启 curses 模式.

相对的, 在结束 curses 模式前 (通常在您结束程式前) 也必须以
endwin()来关闭 curses 模式.

```
#include

main()
{
initscr();
: :
: :
: :
endwin();
}
```

这是一般 curses 程式标准的模式.

此外, 您可以就您程式所须, 而做不同的设定. 当然, 您可以不做设定,而
只是呼叫 initscr().

您可以自己写一个函式来存放所有您所须要的设定. 平常使用时, 只要呼
叫这个函式即可启动 curses 并完成一切设定.

下面的例子, 即是笔者将平常较常用的一些设定放在一个叫 initial()的函

式内.

```
void initial()
{
initscr();
cbreak();
nonl();
noecho();
intrflush(stdscr,FALSE);
keypad(stdscr,TRUE);
refresh();
}
```

各函式分别介绍如下:

□ initscr()

initscr() 是一般 curses 程式必须先呼叫的函数, 一但这个函数被呼叫之後, 系统将根据终端机的形态并启动 curses 模式.

□ endwin()

curses 通常以呼叫 endwin() 来结束程式. endwin() 可用来关闭 curses 模式, 或是暂时的跳离 curses 模式.如果您在程式中须要 call shell (如呼叫 system() 函式) 或是需要做 system call, 就必须先以 endwin() 暂时跳离 curses 模式. 最後再以 wrefresh() doupdate() 来重返 curses 模式.

□ cbreak()
nocbreak()

当 cbreak 模式被开启後, 除了 DELETE 或 CTRL 等仍被视为特殊控制字元外一切输入的字元将立刻被一一读取.当处於 nocbreak 模式时, 从键盘输入的字元将被储存在 buffer 里直到输入 RETURN 或 NEWLINE.在较旧版的 curses 须呼叫 crmode(),nocrmode() 来取代 cbreak(),nocbreak()

□ nl()
nonl()

用来决定当输入资料时, 按下 RETURN 键是否被对应为 NEWLINE 字元 (如 \n).
而输出资料时, NEWLINE 字元是否被对应为 RETURN 和 LINDFEED
系统预设是开启的.

□ echo()
noecho()

此函式用来控制从键盘输入字元时是否将字元显示在终端机上.系统预设是开启的.

□ intrflush(win,bf)

呼叫 intrflush 时须传入两个值:
win 为一 WINDOW 型态指标, 通常传入标准输出入萤幕 stdscr
bf 为 TRUE 或 FALSE

当 bf 为 true 时, 当输入中断字元 (如 break) 时, 中断的反应将较为快速.但可能会造成萤幕的错乱.

□ keypad(win,bf)

呼叫 keypad 时须传入两个值:
win 为一 WINDOW 型态指标, 通常传入标准输出入萤幕 stdscr
bf 为 TRUE 或 FALSE

当开启 keypad 後, 可以使用键盘上的一些特殊字元, 如上下左右等方向键, curses 会将这些特殊字元转换成 curses.h 内定义的一些特殊键. 这些定义的特殊键通常以 KEY_ 开头.

□ refresh()

refresh() 为 curses 最常呼叫的一个函式.

curses 为了使萤幕输出入达最佳化, 当您呼叫萤幕输出函式企图改变萤幕上的画面时, curses 并不会立刻对萤幕做改变, 而是等到 refresh() 呼叫後, 才将刚才所做的变动一次完成. 其馀的资料将维持不变. 以尽可能送最少的字元至萤幕上. 减少萤幕重绘的时间. 如果是 initscr() 後第一次呼叫 refresh(), curses 将做清除萤幕的工作.

■ 游标的控制

move(y,x) 将游标移动至 x,y 的位置

getyx(win,y,x) 得到目前游标的位置
(请注意! 是 y,x 而不是 &y,&x)

■ 有关清除萤幕的函式

clear()
erase() 将整个萤幕清除
(请注意配合refresh() 使用)

■ 如何在萤幕上显示字元

echochar(ch) 显示某个字元

addch(ch) 显示某个字元
mvaddch(y,x,ch) 在(x,y) 上显示某个字元
相当於呼叫 move(y,x);addch(ch);

addstr(str) 显示一串字串
mvaddstr(y,x,str) 在(x,y) 上显示一串字串
相当於呼叫 move(y,x);addstr(str);

printw(format,str) 类似 printf() , 以一定的格式输出至萤幕
mvprintw(y,x,format,str) 在(x,y) 位置上做 printw 的工作.
相当於呼叫 move(y,x);printw(format,str);

■ 如何从键盘上读取字元

getch() 从键盘读取一个字元 (注意! 传回的是
整数值)
getstr() 从键盘读取一串字元
scanw(format,&arg1,&arg2...) 如同 scanf, 从键盘读取一串字元

□例:

```
int ch;
char string1[80]; /* 请注意! 不可宣告为 char *string1; */
char string2[80];

echo(); /* 开启 echo 模式, 使输入立刻显示在萤幕上 */
ch=getch();
string1=getstr();
scanw("%s",string2);
mvprintw(10,10,"String1=%s",string1);
mvprintw(11,10,"String2=%s",string2);
```

■ 如何利用方向键

curses 将一些如方向键等特殊控制字元, 以 KEY_ 为开头定义在 curses.h

这个档案里头, 如 KEY_UP 即代表方向键的 ″ ↑ ″. 但, 如果您想使用

curses.h 所为您定义的这些特殊键的话, 您就必须将 keypad 设定为

TRUE. 否则, 您就必须自己为所有的特殊键定义了.

curses.h 为一些特殊键的定义如下:

KEY_UP 0403 ↑
KEY_DOWN 0402 ↓
KEY_LEFT 0404 ←
KEY_RIGHT 0405 →
KEY_HOME 0406 Home key (upward+left arrow)
KEY_BACKSPACE 0407 backspace (unreliable)
KEY_F0 0410 Function keys.
KEY_F(n) (KEY_F0+(n)) formula for f .
KEY_NPAGE 0522 Next page
KEY_PPAGE 0523 Previous page

以上仅列出笔者较常使用的一些控制键, 至於其他控制键的定义, 请自行参

阅 man curses (* 注三)

一并为您列出其他常用的一些特殊字元

[TAB] /t
[ENTER] /r
[ESC] 27
[BACKSPACE] 127

■ 如何改变萤幕显示字元的属性

为了使输出的萤幕画面更为生动美丽, 我们常须要在萤幕上做一些如反白,

闪烁等变化. curses 定义了一些特殊的属性, 透过这些定义, 我们也可以

在 curses 程式□控制萤幕的输出变化.

attron(mod) 开启属性
attroff(mod) 关闭属性

curses.h 里头定义了一些属性, 如:

A_UNDERLINE 加底线
A_REVERSE 反白
A_BLINK 闪烁
A_BOLD 高亮度
A_NORMAL 标准模式 (只能配合 attrset() 使用)

当使用 attron() 开启某一种特殊属性模式後, 接下来在萤幕的输出都会以

该种属性出现. 直到您呼叫 attroff() 将此模式关闭.

请注意, 当您欲 attron() 开启另一种属性时, 请记得利用 attroff()先关

闭原来的属性, 或直接以 attrset(A_NORMAL) 将所有特殊属性关闭.否则,

curses 会将两种属性做重叠处理.

□例:

```
attrset(A_NORMAL); /* 先将属性设定为正常模式 */

attron(A_UNDERLINE); /* 加底线 */

mvaddstr(9,10,"加底线"); /* 加底线输出一串字元 */

attroff(A_UNDERLINE); /* 关闭加底线模式, 恢复正常模式 */

attron(A_REVERSE); /* 开启反白模式 */

mvaddstr(10,10,"反白"); /* 输出一串反白字元 */

attroff(A_REVERSE); /* 关闭反白模式, 恢复正常模式 */

attron(A_BLINK); /* 开启闪烁模式 */

mvaddstr(11,10,"闪烁"); /* 输出一串闪烁字元 */

attroff(A_BLINK); /* 关闭闪烁模式, 恢复正常模式 */

attron(A_BOLD); /* 开启高亮度模式 */

mvaddstr(12,10,"高亮度"); /* 输出一串高亮度字元 */

attroff(A_BOLD); /* 关闭高亮度模式, 恢复正常模式 */
```

■ 其他常用的一些函式

beep() 发出一声哔声
box(win,ch1,ch2) 自动画方框 ch1: 画方框时垂直方向所用字元
ch2: 画方框时水平方向所用字元

example: box(stdscr,'|','-');
将以 | 及 - 围成一个方框

■ 应用完整□例

下面所举的例子, 即完全利用刚刚所介绍的含式来完成.这个程式可将从键

盘上读取的字元显示在萤幕上, 并且可以上下左右方向键来控制游标的位置

, 当按下 [ESC] 後, 程式即结束.

您有没有发现, 这不就是一个简单全萤幕编辑器的雏形吗?

```
#include /* 引进 curses.h , 并自动引进
stdio.h */


#define StartX 1 /* 决定游标初始位置 */
#define StartY 1

void initial();

main()
{
int x=StartX; /* 宣告 x,y 并设定其初值
*/

int y=StartY;
int ch; /* 宣告 ch 为整数,配合 getch()
使用 */


initial(); /* 呼叫 initial(), 启动 curses
模式, */

/* 并完成其它设定
*/


box(stdscr,'|','-'); /* 画方框
*/
```

```
attron(A_REVERSE); /* 开启反白模式
*/

mvaddstr(0,20,"Curses Program"); /* 在 (20,0) 处输出反白字元
*/

attroff(A_REVERSE); /* 关闭反白模式
*/


move(x,y); /* 将游标移至初始位置
*/


do { /* 以无限回圈不断等待输入
*/

ch=getch(); /* 等待自键盘输入字元
switch(ch) { /* 判断输入字元为何
*/


case KEY_UP: --y; /* 判断是否"↑"键被按下
*/

break;
case KEY_DOWN: ++y; /* 判断是否"↓"键被按下
*/

break;
case KEY_RIGHT: ++x; /* 判断是否"→"键被按下
*/

break;
case KEY_LEFT: --x; /* 判断是否"←"键被按下
*/

break;
case '\r': /* 判断是否 ENTER 键被按下
*/

++y;
x=0;
break;
case '\t': /* 判断是否 TAB 键被按下
*/
```

```
x+=7;
break;
case 127: /* 判断是否 BACKSPACE 键被按下
*/

mvaddch(y,--x,' ');/* delete 一个字元
*/

break;

case 27: endwin(); /* 判断是否[ESC]键被按下
*/

exit(1); /* 结束 curses 模式
*/

/* 结束此程式
*/


default:
addch(ch); /* 如果不是特殊字元, 将此字元印
出 */

x++;
break;
}
move(y,x); /* 移动游标至现在位置
*/

} while (1);
}

void initial() /* 自定开启 curses 函式
*/

{
initscr();
cbreak();
nonl();
noecho();
intrflush(stdscr,FALSE);
keypad(stdscr,TRUE);
refresh();
}
```

■ 後记

学完了上述的一些命令，相不相信您已经可以写出一个漂亮的全萤幕编辑器了？事实上，curses 提供的函式不下 200 个，可是笔者认为，一切再复杂的函式都可以用本文提到的一些组合变化而成，学了太多的函式，只是徒增自己困扰罢了．当然，如果您对其它函式有兴趣，可以自行参阅 curses 说明档．（ 方法：% man curses ）本文不过行抛砖引玉之效，也希望未来能陆续出现更多同学自行创作的程式．

* 任何疑问及建议，欢迎 e-mail 至 ljh@CCCA.NCTU.edu.tw. 谢谢 ！ *

注一：
请参考 /usr/share/lib/termcup
/usr/share/lib/terminfo/s/sun

注二：
1.如果是 BSD 的版本，需使用
cc [file.c] -lcurses -ltermcap 来完成 compile.
2.计中工作站不知何故将原来的 /usr/5bin/cc 更改为 /usr/5bin/cc.org

因此，您若想在计中工作站 compile curses 程式.需以 /usr/5bin/cc.org 取代 /usr/5bin/cc ，否则 compile 可能发生错误.

3.较旧版的 curses 需同时引进 curses 和 termlib 这两个 library，因此，您必须使用 /usr/5bin/cc [file.c] -lcurses -ltermlib 来 compile.

注三：
根据笔者的经验，上下左右方向键应可正常使用而不会发生问题，但其它如 PgUp,PgDn,功能键,Home,End 等特殊键，很容易
无法使用，因此，若您的程式须要在不同的机器上使用，建议您只用方向键来控制，其它的特殊键少用为妙.
至於 PgUp,PgDn 一些特殊键的控制方法，由於较为复杂，有兴趣的同学可参考 tin 原始程式 curses.c 内所使用的一些方
--

● 上一篇教程： UNIX系统编程常用库函数说明 **我要纠错 交流讨论** ● 下一篇教程