MANUAL DE INICIACIÓN A LA LIBRERÍA NCURSES:

Para instalar las librerías y actualizar otros componentes a través de internet, debe abrirse una terminal en modo superusuario y escribir:

 # apt-get update
 # apt-get install make gcc anjuta ncurses-bin libncurses5-dev

ncurses para remolones

Por **Carles Pina i Estany**
http://bulma.net/body.phtml?nIdNoticia=2004

Las ncurses son unas librerías que ayudan a la programación en modo texto para Linux. Es decir, movimientos de cursores, colores, recoger teclas (sin tener que esperar el "Intro") y un largo etcétera. Lo que hacia conio.h en otros sistemas.

También hay gente que a veces tiene problemas con printf y scanf/fgets, debido a buffers intermedios de Linux. Con las ncurses ya no habrá este problema.

En el escrito en cuestión veremos como empezar a programar con ncurses de forma práctica.

Introducción

Muchos habíamos usado la librería `conio.h` con el BorlandC o otros compiladores. En Linux existe una librería que intenta "emularla", llamada uConio pero no es estándard de Unix ni hay paquete para Debian, etc.

En lugar de usar la librería uConio, podemos usar las ncurses. Estas pueden ser usadas desde una forma muy simple hasta para hacer auténticas maravillas. En este artículo veremos una introducción práctica para empezar a usarlas.

Existe bastante documentación por Internet, pero suele ser demasiado profunda al acercarnos por primera vez a la programación usando esas librerías. Incluso, en muchas ocasiones no necesitamos muchas de las funcionalidades que nos ofrecen.

Para compilar programas que usen las ncurses en Debian tendremos que instalar el paquete `libncurses5-dev`. Seguramente lo tendremos instalado ya que también se necesita para compilar el Kernel.
En otras distribuciones necesitaremos instalar un paquete de nombre parecido.

Preparativos

Un programa de ncurses tendrá dos cosas significativas:

- Tendrá este include: `#include <ncurses.h>`
- Se compilará de una forma parecida a `gcc -lncurses fichero.c`

Programando con ncurses

El "Hola mundo" con ncurses

Probaremos un programa así:

```
#include <ncurses.h>

int main() {
        initscr();
        printw("Hola Bulma!!!");
        refresh();
        getch();
        endwin();
        return 0;
}
```

`initscr`: es para entrar en modo ncurses
`printw`: es para imprimir en la "ventana"*****. Es importante usar printw, scanfw, etc. y **no** usar printf, scanf, ya que tendríamos resultados no deseados.
`refresh`: aquí es donde realmente se refresca la pantalla. Podemos hacer varios printw y al final un solo refresh para actualizar la pantalla. Si no hacemos el refresh quizás no saldrá impreso por pantalla.
`getch`: espera una sola pulsación de una tecla. Nos devuelve el código ASCII de la tecla pulsada como un entero. Fijaros que no espera el "Enter" final de línea
`endwin`: terminamos el modo de ncurses. Si no lo hacemos, nos quedará el terminal medio desconfigurado. Para solucionarlo podríamos hacer `reset` desde la misma consola.

Recordar una vez más de compilar con **`gcc -lncurses programa.c`**

(*)En ncurses podemos gestionar ventanas, escribir en ellas de forma independiente, generar scrolls, etc.. Para hacer una ventana usaríamos la función `newwin`, para eliminarla `delwin`, y un largo etcétera (`man newwin`). En este escrito, no crearemos ventanas, por tanto todo el terminal en sí será considerado una ventana.

Entrada de teclado

Para pedir cosas a un usuario con ncurses haríamos:

```
#include <ncurses.h>
int main () {
        char cadena[128];
        initscr();

        printw("Dime tu nombre\n");
        scanw("%s",cadena);
        printw("Te llamas: %s\n",cadena);

        refresh();
        getch();
        endwin();
        return(0);
}
```

La sintaxis del `scanw` es la misma que la del `scanf` tradicional.

Si no quisiéramos que lo que escribe el usuario saliera por pantalla (sin echo):

```
        printw("Escribe la contraseña: ");
        noecho();
        scanw("%s",cadena);
        echo();
```

```
        printw("La contraseña es: %s\n",cadena);
```

Fácil, ¿no?

"Limpiar" la pantalla

Para "limpiar" la pantalla podemos hacer, sencillamente:

```
        erase();
```

Moviendonos por la pantalla

Si nos queremos posicionar en un sitio de la pantalla usaremos la función `move`:

```
        move (10,2);
        printw("Escribe la contraseña: ");
```

¡Ojo! que es `move(int y, int x);` (fila, columna). Los que habíamos usado la `conio.h` era `gotoxy(int x, int y)` (va al revés).

Igualmente podríamos haber usado:

```
        mvprintw(10,2,"Escribe la contraseña: ");
```

Poniendo colores en la pantalla

Si quisiéramos hacer un `printw` con colores:

```
        if (has_colors()) {
        start_color();

        init_pair(1,COLOR_RED,COLOR_YELLOW);
        attron(COLOR_PAIR(1));
        printw("Escribe la contraseña: ");
        attroff(COLOR_PAIR(1));
        }
```

`has_colors`: devuelve 0 o 1 si el terminal tiene soporte para colores o no.
`start_color`: inicia el modo de colores. Si no lo hacemos no veremos los colores.
`init_pair`: asociamos el "par" 1 el color rojo para el texto y el amarillo para el fondo (los colores disponibles con BLACK, RED, GREEN, YELLOW, BLUE, MAGENTA, CYAN, WHITE).
`attron`: y ponemos los colores de forma activa. También podríamos hacer:

```
        attron(COLOR_PAIR(1) | A_UNDERLINE);
```

Y así lo tendríamos subrallado. Tenemos disponibles `A_NORMAL`,`A_BLINK`, `A_BOLD`, etc. (`man attron` para ver más opciones)

Es más, si queremos se pueden personalizar los colores en sí mediante la función `init_color` (`man init_color`).

Enlaces

- El ncurses Programming HOWTO
- Un FAQ de ncurses
- Otro manual de ncurses
- Escribir Programas con ncurses (en castellano)
- `man ncurses`
- El `man` de cada función de ncurses (muy bien detallados)

Queda mucho para hacer con esas librerías. Por ejemplo toda la gestión de ventanas, crearlas, modificarlas, imprimir dentro; menús, submenús; usar el mouse, etc.
Eso sólo ha sido una manera de introducirnos en la programación de ncurses, a partir de aquí podemos hacer si lo queremos muchas otras cosas.

Comentar que hay un port para Perl de las ncurses (paquete `libcurses-perl`) y, también desde Perl, una manera rápida de hacer ventanas y formularios usando las libcurses (paquete `libcurses-widgets-perl`)

Programación con Ncurses

Por Wilson Libardo Pantoja Y.
http://gluc.unicauca.edu.co/wiki/index.php/Programaci%F3n_con_Ncurses

Las ncurses son librerías que permiten trabajar funciones similares a las contenidas en la librería conio.h del viejo turboC de Borland, entre las cuales están: gotoxy(), clrscr(), textcolor(), getch(), etc. Con ncurses se pueden hacer aplicaciones gráficas elegantes en modo texto y de forma muy fácil.

En Internet he encontrado mucha documentación sobre ncurses, sin embargo, para un principiante, la información es tan profunda que resulta complicado su entendimiento. Mi intención es brindar un documento sencillo e introductorio sobre esta librería.

Un Programa sencillo: “Hola linux..”

Utiliza un editor de texto cualquiera y copia (o digita) este programa y grábalo con el nombre programa.c:

```
#include <ncurses.h>
int main() {
       initscr();
       printw("Hola linux!");
       refresh();
       getch();
       endwin();
       return 0;
}
```

Nótese que en este programa se ha incluído: #include <ncurses.h> , con lo cual se tiene acceso al uso de las siguientes funciones:

- initscr(): se la utiliza para inicializar un ventana en modo ncurses.

- printw(): es para imprimir texto en la ventana. Es importante usar printw y scanfw en lugar de printf, scanf, porque se tendrían salidas erróneas debido a los buffers intermedios de linux.

- refresh(): sirve para refrescar la pantalla, todos los printw se muestran efectivamente al hacer refresh.

- getch(): espera una sola pulsación de una sola tecla (sin tener que presionar “Enter”), devolviendo el código ASCII de la tecla pulsada como un entero.

- endwin(): finaliza el modo de ncurses. Es importante colocar esta instrucción, de lo contrario el terminal queda desconfigurado, obligando a cerrar la consola.

Compilar el programa “Hola Linux”

Para compilar desde consola este programa se debe utilizar el parámetro –lncurses, que indica al compilador que el programa está utilizando curses; el comando completo es:

gcc -lncurses programa.c

Ejemplos

A continuación he elaborado algunos ejemplos sencillos y prácticos que ilustran como trabajar curses.

```
/*dibujar_cuadro.c
ESTE PROGRAMA DIBUJA UN CUADRO. UTILIZA LA LIBRERIA curses.h QUE SUSTITUYE DE ALGUNA
FORMAA LA      CONIO.H DEL VIEJO TURBO C
COMPILE ESTE PROGRAMA CON LA INSTRUCCION:  gcc dibujar_cuadro.c -o ejecutable -lncurses*/
#include <curses.h> //Incluiremos una librería a nuestro sistemas
#include <stdio.h>
void salir (void);  //Esta funcion hará que nuestro programa se cierre
int main(void)
{
   int i;char c;
   initscr();   /*Esta función  inicializa  ncurses. Para todos los programas
                  debemos siempre inicializar  ncurses y luego finalizarla, como
                 veremos adelante. */
   c=95;//caracter ascci horizontal
   for(i=1;i<=120;i++)
   {
      //linea horizontal superior
      move(1,i);  //Aqui estamos moviendo el cursor para a linea 1 columna i.
      printw("%c",c);  //Imprimimos un texto en la posición establecida.
      //linea horizontal inferior
      move(40,i);  //Aqui estamos moviendo el cursor para a linea 40 columna i.
      printw("%c",c);  //Imprimimos un  texto en la posición establecida.
   }
   c=124 ; //caracter ascci vertical
   for(i=2;i<=40;i++)
```

```
   {
      //linea vertical izquierda
      move(i,1);
      printw("%c",c);
      //linea vertical derecha
      move(i,120);
      printw("%c",c);
   }
   refresh();
   //getch(); //si se desea hasta que se pulse una tecla
   sleep(3);//se detiene tres segundo
   salir();    // Salir del programa
}
/*****************************************************************/
void salir()
{
   endwin(); /*Siempre que finalizamos un programa con una biblioteca curses,
                  debemos ejecutar este comando.*/
   exit(0);
}
```

A continuación un segundo ejemplo:

```
/*Este ejemplo captura las teclas y muestra el valor devuelto
por cada uno de ellas. La tecla <Esc> parece más lenta pero eso
es debido a que existe la necesidad de distinguir un caracter <Esc>
aislado de una secuencia de caracteres que empiece por <Esc> y que
podría venir de una tecla de función por ejemplo.*/
#include <stdio.h>
#include <stdlib.h>
#include <curses.h>
WINDOW *win; /** manejaremos una única ventana de pantalla completa **/
/*****************************************************************/
void IniVideo(){
       win=initscr(); /* Crea la ventana */
       clear();  /* Borra la pantalla entera bajo ncurses */
       refresh(); /* Actualiza la ventana con los cambios */
       noecho();
       cbreak();
       keypad(win, TRUE);
}
/************************/
Exit(){
       refresh();
       endwin();
       exit(1);
}
/*****************************************************************/
main () {
       IniVideo();
       move(7, 30); /* x , y */
       printw("Ctrl-C  para terminar");
       for(;;){
              move(12, 30); /* x , y */
              printw("%3d", getch());
              refresh();
       }
       Exit();
}
```

Ahora un tercer ejemplo, bastante elegante porque utiliza colores.

```
#include <curses.h>
#include <stdio.h>
void sair (void);  //Esta función se utiliza al Salir del programa
int main(void)
{
   initscr();   /*Esta función inicializa los ncurses
   start_color(); //Esta función inicia los colores
//Define pares de colores que serán definidos en el programa
init_pair(1,COLOR_WHITE,COLOR_BLUE); //Texto(Blanco) | Fondo(Azul)
   init_pair(2,COLOR_BLUE,COLOR_WHITE); //Texto(Azul) | Fondo(Branco)
   init_pair(3,COLOR_RED,COLOR_WHITE);  //Texto(rojo) | Fundo(Blanco)
   bkgd(COLOR_PAIR(1));  /*Aqui define el color de fondo del programa
   attron(COLOR_PAIR(3));
   move(2,1);  //Aqui mueve el cursor a linea 2 columna 1.
   printw("Olá mundo!!!");  //Imprimimos el texto en la posición especificada
                              en la linea anterior.
   attroff(COLOR_PAIR(3));  /*Esta alterando el par de colores por omisión*/
   attron(COLOR_PAIR(2));
   move(3,1);
   printw("Cualquier tecla para salir"); /*Imprime el texto en la posición
                                           especificada en la línea anterior */
   attroff(COLOR_PAIR(2));
   refresh();    //Actualiza la ventana
   getch();      //Espera que el usuario presione un tecla
   sair();    // llama a la función salir
}
/*****************************************************************/
void salir()
{
   endwin(); /*Siempre que finalizamos un programa con una biblioteca curses,
                    debemos ejecutar este comando.*/
   exit(0);
}
```

Y si aparte de esto todavía quieres trabajar con la conio.h que funcionaba con el Turbo C de DOS, recomiendo esta página en la cual se implementa dicha función a partir de ncurses:

- http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=912&pagina=1&PHPSESSID=61c44ea8680ded9e8e83ba458073cd81

Con lo descrito en este artículo entenderás lo básico para programar con curses en Linux. Ahora, si ya quieres profundizar más recomiendo visitar las siguientes páginas:

- http://lists.debian.org/debian-user-spanish/2001/04/msg00520.html=>Ejemplos sencillos y prácticos
- http://www.hackemate.com.ar/ezines/disidents/disidents004/disidents004/0X10.txt => Ejemplos con curses.
- http://sai.azc.uam.mx/apoyodidactico/ps/Apendice1/psape1.html=>Mucha teoría y algunos ejemplos
- http://ditec.um.es/~piernas/manpages-es/otros/tutorial-ncurses.html = > Para avanzados

A conio.h

A *conio.h* que criei utiliza os recursos da biblioteca *ncurses*, portanto trata-se simplesmente de uma certa "tradução" da ncurses para algo semelhante a conio.h que vem com os compiladores do C/C++ para DOS/Windows.

Eis a conio.h para Linux:

```
// conio.h
// CONIO.H UTILIZANDO OS RECURSOS DA BIBLIOTECA NCURSES     //
// -------------------------------------------------     //
//                                                        //
// DESENVOLVIDO POR: JEFFERSON DOS SANTOS FELIX, ABRIL 2004 //
//                                                        //

#ifndef __NCURSES_H
#include <curses.h>
#endif

#define BLACK       0
#define RED         1
#define GREEN       2
#define BROWN       3
#define BLUE        4
#define MAGENTA     5
#define CYAN        6
#define LIGHTGRAY   7
#define DARKGRAY    8
#define LIGHTRED    9
#define LIGHTGREEN  10
#define YELLOW      11
#define LIGHTBLUE   12
#define PINK        13
#define LIGHTCYAN   14
#define WHITE       15

#define DEFAULT_PAIR 57

int initconio(void);
int endconio(void);
int clrscr(void);
int textcolor(short color);
int textbackground(short color);
int gotoxy(int x, int y);
int wherex(void);
int wherey(void);

short cur_pair;
int cur_bold;

int initconio(void)
{
  int f, b;
  short p;
  initscr();
  start_color();
  p = 1;
  for(f = 0; f < 8; f++)
    for(b = 0; b < 8; b++, p++)
      init_pair(p, f%8, b%8);
  cur_pair = DEFAULT_PAIR;
  cur_bold = 0;
  bkgd(COLOR_PAIR(cur_pair));
  color_set(cur_pair, NULL);
  attr_off(A_BOLD, NULL);
  return 0;
}

int endconio(void)
{
  endwin();
  return 0;
}
```

```
int clrscr(void)
{
  bkgd(COLOR_PAIR(cur_pair));
  if(cur_bold == 1)
    attr_on(A_BOLD, NULL);
  else
    attr_off(A_BOLD, NULL);
  clear();
  return 0;
}

int textcolor(short color)
{
  short f, b, x, y;
  short p;
  pair_content(cur_pair, &f, &b);
  p = 1;
  for(x = 0; x < 8; x++)
    for(y = 0; y < 8; y++, p++)
      if((x == (color%8))&&(y == b))
        cur_pair = p;
  color_set(cur_pair, NULL);
  if(color >= 8)
  {
    cur_bold = 1;
    attr_on(A_BOLD, NULL);
  }
  else
  {
    cur_bold = 0;
    attr_off(A_BOLD, NULL);
  }
  return 0;
}

int textbackground(short color)
{
  short f, b, x, y;
  short p;
  pair_content(cur_pair, &f, &b);
  p = 1;
  for(x = 0; x < 8; x++)
    for(y = 0; y < 8; y++, p++)
      if((x == f)&&(y == (color%8)))
        cur_pair = p;
  color_set(cur_pair, NULL);
  return 0;
}

int gotoxy(int x, int y)
{
   move(x - 1, y - 1);
   return 0;
}

int wherex(void)
{
   int x, y;
   getyx(stdscr, x, y);
   return x + 1;
}

int wherey(void)
{
```

```
   int x, y;
   getyx(stdscr, x, y);
   return y + 1;
}
```

Introducción a Ncurses

por Reha K. Gerçeker
http://es.tldp.org/LinuxFocus/pub/mirror/LinuxFocus/Castellano/March2002/article233.shtml

Resumen:

Ncurses es una librería que proporciona mapeado de las teclas de función, funciones de dibujado en pantalla y la habilidad de usar múltiples ventanas no solapadas en terminales de texto.

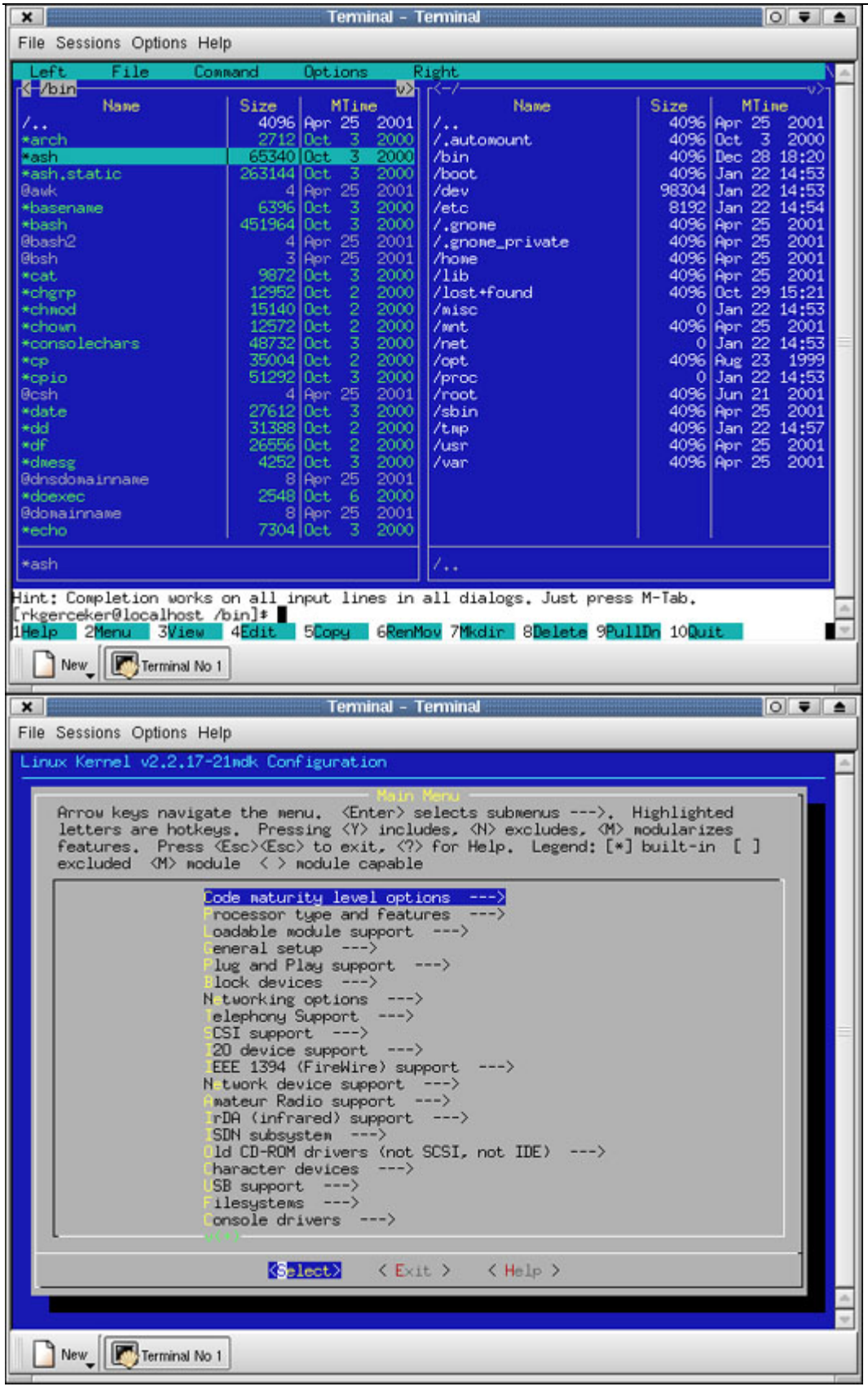
¿Qué es Ncurses?

¿Quiere que sus programas tengan una colorida interfaz basada en el terminal? Ncurses es una librería que proporciona funcionalidad de ventanas para terminales de texto. Algunas cosas que ncurses es capaz de hacer son:

- Usar la pantalla completa como quieras.
- Crear y controlar ventanas.
- Usar 8 colores diferentes.
- Darle a su programa soporte para el ratón.
- Usar las teclas de función del teclado.

Es posible usar ncurses en cualquier sistema Unix que sigua la norma ANSI/POSIX. Aparte de esto la librería es capaz de detectar las propiedades del terminal de la base de datos del sistema y actuar en consecuencia, proporcionando una interfaz independiente del terminal. Por lo tanto, ncurses puede ser usado con garantías para diseños que vayan a trabajar en diferentes plataformas y terminales..

Midnight Commander es un ejemplo de un programa que utiliza ncurses. También la interfaz utilizada para la configuración del kernel está escrita con ncurses. Más abajo se pueden ver unas capturas de pantalla de estos ejemplos.



¿Donde descargarlo?

Ncurses está desarrollado bajo GNU/Linux. Para descargar la última versión, ver información detallada y encontrar otros enlaces, visite www.gnu.org/software/ncurses/.

Lo básico

Para utilizar la librería, debe incluir curses.h en su código y asegurarse de enlazar su código con la librería curses. Eso se hace pasando el parámetro -lcurses a gcc.

Es necesario tener algún conocimiento acerca de la estructura básica de datos cuando se trabaja con ncurses. Esta es la estructura WINDOW y, como indica su nombre, es usada para representar las ventanas que se creen. Casi todas las funciones de la librería tienen un puntero a WINDOW como parámetro.

Los componentes más usados de ncurses son las ventanas. Incluso si no crea sus propias ventanas, la pantalla es considerada como una ventana. Así como el descriptor FILE stdout de la librería estándar de E/S representa la pantalla (cuando no hay redirecciones), ncurses tiene el puntero de tipo WINDOW stdscr, otro puntero de tipo WINDOW llamado curscr también es definido en la librería. Mientras stdscr representa la pantalla, curscr representa la pantalla actual conocida por la librería. Aquí se preguntará: "¿Cuál es la diferencia?". Siga leyendo.

Para poder usar las funciones y variables de ncurses, primero tiene que llamar a la función initscr. Esta función asigna memoria para variables tales como stdscr o curscr y prepara la librería para ser usada. En otras palabras, todas las funciones de ncurses tienen que ir después de la llamada a initscr. Asímismo, debe llamar a endwin una vez que haya acabado con ncurses. Esto libera la memoria usada por ncurses. Después de llamar a endwin no podrá usar ninguna función de ncurses a menos que vuelva a llamar a initscr.

Entre las llamadas a initscr y endwin, asegúrese de no enviar la salida a pantalla usando las funciones de la librería estándar de E/S. Si no, es probable que la salida en pantalla se corrompa y no se parezca en nada a lo que quería. Cuando ncurses está activo, use únicamente sus funciones para enviar la salida a pantalla. Antes de llamar a initscr o después de llamar a endwin puede hacer lo que desee.

Actualizando la pantalla: refresh

La estructura WINDOW no solamente guarda la altura, anchura y posición de la ventana, sino que también almacena su contenido. Cuando usted escribe en una ventana, los contenidos de la ventana son cambiados, pero eso no significa que esos cambios aparezcan en la pantalla inmediatamente. Para que los cambios se muestren en pantalla hay que llamar a las funciones refresh o wrefresh.

Aquí está la diferencia entre stdscr y curscr. Mientras que curscr almacena el contenido de la pantalla actual, stdscr puede tener una información diferente después de las llamadas a las funciones de salida de ncurses. Si desea que los últimos cambios sobre stdscr sean volcados en curscr, entonces tendrá que llamar a la función refresh. En otras palabras, refresh es la única función que trata con con curscr. Es recomendable que no se líe con curscr y deje que sea la función refresh quien se encargue de actualizarlo.

refresh tiene un mecanismo para actualizar la pantalla lo más rápidamente posible. Cuando la función es llamada, solamente actualiza las líneas de la ventana que hayan sido cambiadas. Esto ahorra tiempo de CPU y evita al programa tener que volver a escribir la misma información en pantalla.. Este mecanismo es la razón por la que las funciones de ncurses y las funciones de E/S estándar pueden producir resultados erróneos cuando son usadas conjuntamente. Cuando las funciones de ncurses son llamadas éstas fijan un indicador que indica a refresh cuáles son las líneas que han cambiado, nada de ésto ocurre cuando se llama a las funciones de la librerías de E/S estándar.

refresh y wrefresh básicamente hacen lo mismo. wrefresh toma un puntero a WINDOW como parámetro y únicamente actualiza el contenido de ésta ventana. refresh() es equivalente a wrefresh(stdscr). Como se verá más tarde, al igual que wrefresh, la mayoría de las funciones de ncurses tienen macros que aplican estas funciones a stdscr.

Creando nuevas ventanas

Hablemos ahora un poco de subwin y newwin, las funciones para crear nuevas ventanas. Estas funciones toman la altura, el ancho y las coordenadas de la esquina superior izquierda de la nueva ventana como parámetros y devuelven un puntero de tipo WINDOW que apunta a su nueva ventana. Este puntero puede ser usado por wrefresh y otras funciones que se verán más adelante.

"¿Si hacen lo mismo, por qué duplicar las funciones?" se puede preguntar usted. Está en lo cierto, hay pequeñas diferencias, subwin crea una nueva ventana que es subventana de otra. Una ventana creada de esta forma hereda las propiedades de la ventana padre. Estas propiedades pueden ser cambiadas más adelante sin afectar a la ventana padre.

Aparte de esto, hay una cosa que las mantiene unidas. La matriz de caracteres que almacena los contenidos de la ventana son compartidos entre las ventanas padre e hijo. En otras palabras, los caracteres que estén en la intersección de las dos ventanas pueden ser cambiados por cualquiera de ellas. Si la ventana padre escribe en dicho área, el contenido de la ventana hija también cambia, y a también la inversa.

Al contrario que subwin, newwin crea una ventana totalmente nueva. Esa ventana, a menos que tenga sus propias subventanas, no compartirá su matriz de caracteres con ninguna otra ventana. La ventaja de usar subwin es que al usar una matriz de caracteres compartida el gasto de memoria es menor. Por otro lado, cuando las ventanas empiezan a escribirse unas sobre otras, el uso de newwin tiene sus propias ventajas.

Puede crear sus propias subventanas hasta cualquier profundidad. Cada subventana puede tener a su vez sus propias subventanas, pero entoces tiene que tener en cuenta que la matriz de caracteres será compartida por más de dos ventanas.

Cuando haya acabado con la ventana que ha creado, puede borrarla con la función delwin. Le aconsejo que consulte las páginas del man para ver la lista de parámetros de estás funciones.

Escribir a Ventanas, Leer de Ventanas

Hemos hablado acerca de stdscr, curscr, refrescar la pantalla y crear nuevas ventanas. ¿Pero que pasa cuando escribimos en una ventana? ¿O cuando leemos datos de una ventana?

Las funciones usadas para ello son similares a sus contrapartidas de la librería de E/S estándar. Entre estas funciones está printw en lugar de printf, scanw en lugar de scanf, addch en lugar de putc o putchar, getch en lugar de getc o getchar. Estas funciones se usan de la forma usual, sólo sus nombres son diferentes. De forma similar, addstr puede ser utilizada para escribir una cadena en una ventana y getstr para leer una cadena de una ventana. Todas esta funciones, con la letra 'w' al principio de su nombre y un puntero a WINDOW como primer parámetro hacen su trabajo en una ventana diferente a stdscr. Por ejemplo, printw (...) y wprintw(stdscr, ...) son equivalentes, así como refresh() y wrefresh(stdscr).

Entrar en detalles con estas funciones llevaría mucho tiempo. Las páginas del man son la mejor fuente de información para aprender acerca de sus descripciones, prototipos, valores de retorno y otras notas. Le aconsejo que las mire para cada función que use. Ofrecen detallada y valiosa información. La última sección de este artículo en la que expongo un programa de ejemplo también puede servir de tutorial sobre como usar estas funciones.

Cursores Físicos y Lógicos

Es necesario explicar como funcionan los cursores físicos y lógicos después de hablar acerca de cómo escribir y leer en ventanas. Lo que se entiende por cursor físico es el típico cursor parpadeante que se ve en pantalla, únicamente puede haber un cursor físico. Por otro lado, los cursores lógicos pertenecen a las ventanas de ncurses y cada ventana tiene el suyo propio. Así pues puede haber muchos cursores lógicos.

El cursor lógico está en la posición de la ventana donde el proceso de lectura o escritura va a comenzar. Por lo tanto, moviendo el cursor lógico podremos escribir en cualquier punto de la pantalla o ventana cuando queramos. Esta es una de las ventajas de ncurses sobre las librería de E/S estándar.

La función que se encarga de mover el cursor lógico es move o, como podrá deducir fácilmente, wmove. move es una macro de wmove, hecha para stdscr.

Otros asunto es la coordinación entre los cursores físicos y lógicos. La posición del cursor físico después de un proceso de escritura depende del flag _leave, presente en la estructura WINDOW. Si _leave está activo, el cursor lógico se moverá hasta la posición del cursor físico después de realizar la escritura (donde se escribió el último caracter). Si _leave no está activo, el cursor físico vuelve a la posición del cursor lógico después de realizar la escritura (donde se escribió el primer caracter). El flag _leave es controlado por la función leaveok.

La función que mueve el cursor físico es mvcur. Al contrario que otras, mvcur tiene efecto inmediatamente, antes incluso de la siguiente llamada a refresh. Si desea que el cursor físico sea invisible use la función curs_set. En las páginas del man encontrará más detalles.

También hay macros que combinan las funciones de movimiento y escritura descritas anteriormente en una simple llamada. Esto está bien explicado en las mismas páginas del man que las funciones addch, addstr, printw, getch, getstr, scanw, etc...

Borrando en Ventanas

Ya sabemos como se puede escribir en ventanas. ¿Pero ahora cómo borramos ventanas, líneas o caracteres?

Borrar, en ncurses, significa rellenar el carácter, la línea o el contenido de la ventana con espacios en blanco. Las funciones que voy a explicar más abajo rellenan los caracteres (posiciones en la pantalla en realidad) necesarios con espacios en blanco y así borran la pantalla.

Primero hablemos de las funciones que se refieren al borrado de un caracter o una línea. Las funciones delch y wdelch borran el caracter que está bajo el cursor lógico de la ventana y desplaza los caracteres que le siguen en la misma línea hacia la derecha. deleteln y wdeleteln borran la línea en la que está el cursor lógico y desplazan hacia arriba todas las líneas que estén más abajo.

Las funciones clroeol y wclroeol borrarán todos los caracteres de la misma línea que estén a la derecha del cursor lógico. clrobot y wclrobot primero llaman a wclrtoeol para borrar todos los caracteres a la derecha del cursor lógico y después borra todas las líneas que haya a continuación.

Aparte de estas, hay funciones que borran la pantalla entera o solamente una ventana. Hay dos métodos para borrar la pantalla entera. El primero es rellenar todas las posiciones con caracteres en blanco y después llamar a la función refresh y la otra es usar el codigo de control integrado en el terminal. El primer método es más lento porque requiere que todas las posiciones de la pantalla sean reescritas una a una mientras que el segundo borra toda la pantalla inmediatamente.

erase y werase rellenan la matriz de caracteres de una ventana con espacios en blanco. En la próxima llamada a refresh la ventana se borrará. Sin embargo, si la ventana que hay que borrar ocupa toda la pantalla, usar estas funciones es una solución poco elegante. Estas funciones usan el método descrito más arriba. Cuando la ventana que hay que borrar ocupa toda la pantalla, es mucho mejor utilizar las siguientes funciones.

Antes de entrar en estas funciones, habría que hablar del indicador _clear. Está en la estructura WINDOW y si está activado, pregunta a refresh para enviar el código de control al terminal cuando es llamado. Al ser llamado, refresh comprueba si la ventana ocupa toda la pantalla (usando el indicador _FULLWIN) y si es así, borra la pantalla con el método integrado en el terminal. Luego sólo tiene que escribir los caracteres y no los espacios en blanco en la pantalla. Esto hace que borrar la pantalla completa sea más rápido. La razón por la que este sistema sólo se usa para ventanas que ocupan toda la pantalla es que el código de control del terminal borra toda la pantalla y no sólo una ventana. El indicado _clear es controlado por la función clearok.

Las funciones clear y wclear son usadas para borrar ventanas que ocupan toda la pantalla. De hecho, estas funciones son equivalentes a hacer una llamada a werase y a clearok. Primero, rellenan la matriz de caracteres de la ventana con espacios en blanco. Y luego, activando el indicador _clear borra la pantalla usando el metodo integrado en el terminal si la ventana ocupa toda la pantalla y en caso contrario lo hace rellenando todas las posiciones con espacios en blanco.

En resumen, si sabe que la ventana a ser borrada ocupa toda la pantalla entonces use clear o wclear. Será más rápido. Si no, no hay ninguna diferencia entre el uso de wclear o werase.

Usando Colores

Los colores que ve en la pantalla se muestran a traves de pares de colores. Esto es así porque cada posición tiene un color de primer plano y otro de segundo plano. Para escribir en color con ncurses tienen que crear sus propios pares de colores y usarlos para escribir en pantalla.

Al igual que initscr necesita ser llamado para poder utilizar ncurses, start_color tiene que ser llamado para poder usar colores. La función necesaria para crear sus propios pares de colores es init_pair. Cuando crea un par de colores con init_pair, éste es asociado con el número que se le pasa como primer parámetro a la función. Así, siempre que quiera usar un par, tiene que referirse a él con el número asignado a COLOR_PAIR.

Aparte de crear pares de colores, necesita algunas funciones para escribir con diferentes pares de colores. Esto se hace con las funciones attron y wattron. Estas funciones hacen que, desde el momento en que son llamadas, todo lo que se escriba en la ventana correspondiente se haga en el par de colores que ha elegido.

También tiene las funciones bkgd y wbkgd que cambia el par de colores asociado con una ventana completa. Cuando alguna de ellas es llamada cambia el color tanto del primer como del segundo plano de todas las

posiciones de la ventana. Esto significa que en la próximo refresco de pantalla, cada posición de la ventana se reescribirá con el nuevo par de colores.

Mire en las páginas del man para saber cuales son los colores disponibles y más detalles de las funciones que hemos mencionado.

Recuadros Alrededor de Ventanas

Puede crear recuadros alrededor de las ventanas para darle un aspecto más agradable a su programa. Hay una macro en la librería llamado box que lo hace por usted. Al contrario que otras funciones wbox no existe, box toma como argumento un puntero de tipo WINDOW.

Puede encontrar fácilmente más detalles de box en las páginas del man. Aunque hay algo más que debería ser mencionado. Ponerle un recuadro a una ventana simplemente consiste en escribir los caracteres necesarios en la matriz de caracteres de la ventana que corresponden con las posiciones de los bordes. Si mas adelante por algún motivo escribe en estas posiciones, el recuadro puede corromperse. Para evitar esto, puede crear una ventana dentro de la ventana original con subwin, poner el [recuadro] en la ventana original y utilizar la ventana interior para escribir.

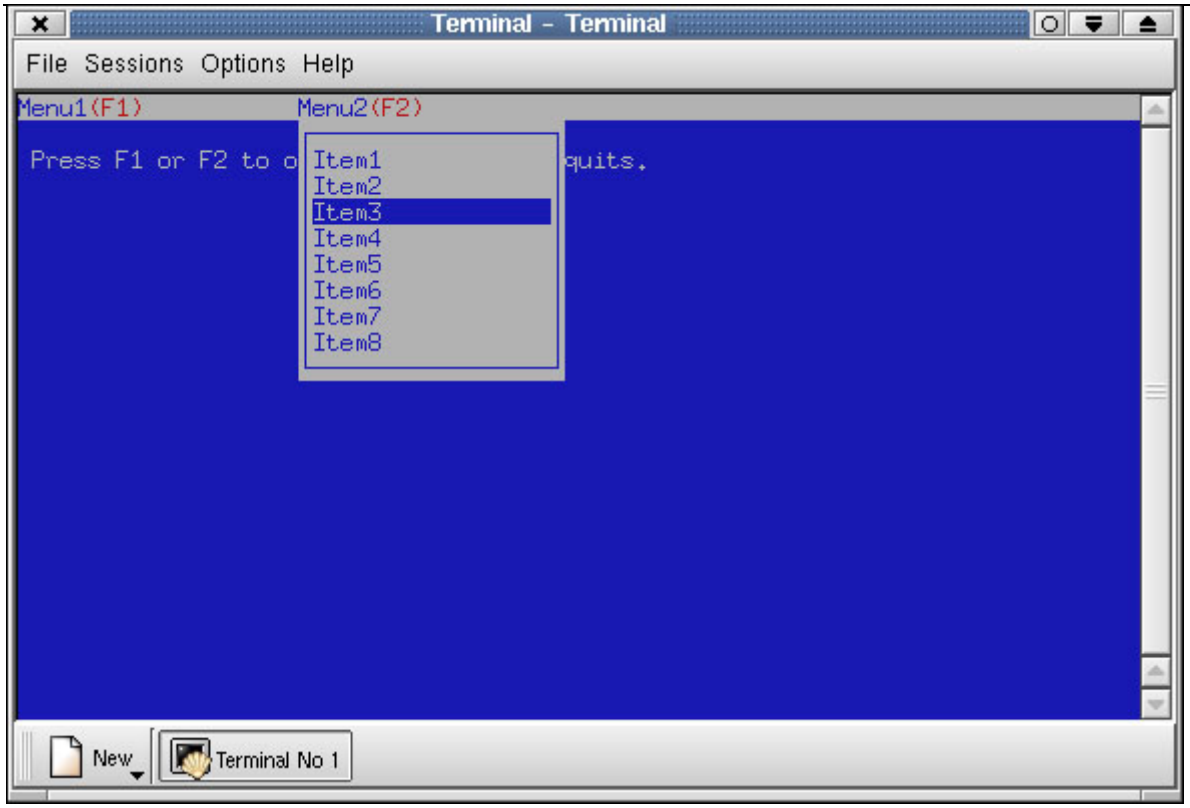
Teclas de Función

Para poder usar las teclas de función, el indicador _use_keypad tiene que ser activado en la ventana desde la que se va a leer la entrada de teclado. keypad es la función que establece el valor de _use_keypad. Una vez activado _use_keypad, puede tomar la entrada de teclado de la forma habitual con las funciones de entrada.

En este caso, si por ejemplo utiliza getch para obtener datos, debe tener cuidado en almacenar los datos en una variable de tipo entero en lugar de una de tipo char. Esto es porque los valores numéricos de las teclas de función son mayores que los valores que puede contener una variable de tipo char. No necesita conocer el valor numérico de las teclas de función ya que la librería proporciona nombres ya definidos para ellas. Estos nombres están listados en la página man de getch.

Un Ejemplo

Vamos a analizar ahora un sencillo y bonito programa. En él los menús van a ser creados usando ncurses y se indicará cuando se selecciones una opción del menú. Un aspecto interesante de este programa es el uso de ventanas ncurses para simular los menús cuando se abren. Lo puede ver en la captura de pantalla de más abajo.



El programa comienza con las cabeceras como es habitual. Luego definimos las constantes para los valores ASCII de las teclas de enter y escape.

```
#include <curses.h>

#include <stdlib.h>

#define ENTER 10
#define ESCAPE 27
```

La siguiente función es las primera que se llama al ejecutar el programa. Primero llama a initscr para inicializar curses y luego a start_color para poder utilizar colores. Los pares de colores que se van a utilizar en el programa se definen a continuación. La llamada a curs_set(0) hace que el cursor físico sea invisible. noecho hace que entrada del teclado no aparezca en pantalla. También se puede usar la función noecho para controlar la entrada del teclado y mostrar únicamente las partes que quiera que se muestren. Para quitar los efectos de la función noecho hay que llamar a la función echo. Finalmente se llama a la función keypad para habilitar las teclas de función cuando se lea la entrada de stdscr. Esto es necesario porque en nuestro programa vamos a utilizar las teclas F1, F2 y los cursores.

```
void init_curses()
{
    initscr();
    start_color();
    init_pair(1,COLOR_WHITE,COLOR_BLUE);
    init_pair(2,COLOR_BLUE,COLOR_WHITE);
    init_pair(3,COLOR_RED,COLOR_WHITE);
    curs_set(0);
    noecho();
    keypad(stdscr,TRUE);
}
```

La siguiente función crea una barra de menú que aparece en la parte de arriba de la pantalla. Si se fija en dicha función se dará cuenta de que la barra de menú que se ve como una línea sencilla en la parte superior en

realidad está definida como una subventana de stdscr con una única línea de alto. Ésta función toma un puntero a esta ventana como parámetro, primero cambia el color de fondo y luego escribe los nombres de los menús. Usamos waddstr para escribir los nombres de los menús. Preste atención a las llamadas a wattron, se usan para escribir con un par de colores diferente (número 3) en lugar de usar el par de colores por defecto (número 2). Recuerde que el par número 2 fue puesto como par por defecto en la primera línea por wbkgd. wattroff es llamado cuando se quiere cambiar al par de color por defecto.

```
void draw_menubar(WINDOW *menubar)
{
    wbkgd(menubar,COLOR_PAIR(2));
    waddstr(menubar,"Menu1");
    wattron(menubar,COLOR_PAIR(3));
    waddstr(menubar,"(F1)");
    wattroff(menubar,COLOR_PAIR(3));
    wmove(menubar,0,20);
    waddstr(menubar,"Menu2");
    wattron(menubar,COLOR_PAIR(3));
    waddstr(menubar,"(F2)");
    wattroff(menubar,COLOR_PAIR(3));
}
```

La siguiente función dibuja los menús cuando las teclas F1 o F2 son presionadas. Para crear el efecto de menú una ventana nueva con el mismo color blanco que la barra de menú es creada sobre la ventana azul que compone el fondo.No queremos que esta nueva ventana sobrescriba los caracteres que hubiera anteriormente en el fondo. Estos deben continuar ahí una vez que el menú sea cerrado. Esto es por lo que la ventana menú no puede ser creada como una subventana de stdscr. Como verá más abajo, la ventana items[0] es creada con la función newwin mientras que las otras ocho ventanas items son creadas como subventanas de items[0]. Aquí items[0] es usada para dibujar una caja alrededor del menú y las otras ventanas items lo son para mostrar los items seleccionados en el menú y también para no sobreescribir los caracteres de la caja alrededor del menú. Para hacer que un item del menú parezca que está seleccionado es suficiente con hacer que el color de fondo sea diferente al del resto de los items. Esto es lo que hace la antepenúltima línea; el color de fondo del primer item es puesto de un color diferente al de los otros y así cuando el menú aparece, el primer item está seleccionado.

```
WINDOW **draw_menu(int start_col)
{
    int i;
    WINDOW **items;
    items=(WINDOW **)malloc(9*sizeof(WINDOW *));

    items[0]=newwin(10,19,1,start_col);
    wbkgd(items[0],COLOR_PAIR(2));
    box(items[0],ACS_VLINE,ACS_HLINE);
    items[1]=subwin(items[0],1,17,2,start_col+1);
    items[2]=subwin(items[0],1,17,3,start_col+1);
    items[3]=subwin(items[0],1,17,4,start_col+1);
    items[4]=subwin(items[0],1,17,5,start_col+1);
    items[5]=subwin(items[0],1,17,6,start_col+1);
    items[6]=subwin(items[0],1,17,7,start_col+1);
    items[7]=subwin(items[0],1,17,8,start_col+1);
    items[8]=subwin(items[0],1,17,9,start_col+1);
    for (i=1;i<9;i++)
        wprintw(items[i],"Item%d",i);
    wbkgd(items[1],COLOR_PAIR(1));
    wrefresh(items[0]);
    return items;
}
```

La función que tenemos a continuación simplemente borra la ventana menú creada por la función anterior. Primero borra las ventanas de los items con delwin y luego libera la memoria asignada para el puntero items.

```
void delete_menu(WINDOW **items,int count)
{
    int i;
    for (i=0;i<count;i++)
        delwin(items[i]);
    free(items);
}
```

La función scroll_menu nos permite movernos entre menús y dentro de ellos. Primero lee con getch las teclas que son pulsadas en el teclado. Si se presionan los cursores de arriba o abajo, entonces se selecciona el item superior o inferior. Esto se hace, como recordará, haciendo que el color de fondo sea diferente que el del resto de los items. Si se presionan los cursores de izquierda o derecha, el menú abierto se cierra y luego se abre el siguiente. Si se presiona la tecla enter, la función devuelve el item seleccionado. Si la tecla presionada es ESC, se cierran todos los menús sin seleccionar ningún item, La función ignora cualquier otra tecla que sea pulsada. En esta función, getch es capaz de leer las teclas de cursor del teclado. Permítame recordarle que esto es posible porque anteriormente, en la función init_curses, se hizo una llamada a keypad (stdscr, TRUE) y el valor devuelto por getch es de tipo int en lugar de ser de tipo cha ya que los valores de las teclas de función son mayores de lo que una variable de tipo char puede contener.

```
int scroll_menu(WINDOW **items,int count,int menu_start_col)
{
    int key;
    int selected=0;
    while (1) {
        key=getch();
        if (key==KEY_DOWN || key==KEY_UP) {
            wbkgd(items[selected+1],COLOR_PAIR(2));
            wnoutrefresh(items[selected+1]);
            if (key==KEY_DOWN) {
                selected=(selected+1) % count;
            } else {
                selected=(selected+count-1) % count;
            }
            wbkgd(items[selected+1],COLOR_PAIR(1));
            wnoutrefresh(items[selected+1]);
            doupdate();
        } else if (key==KEY_LEFT || key==KEY_RIGHT) {
            delete_menu(items,count+1);
            touchwin(stdscr);
            refresh();
            items=draw_menu(20-menu_start_col);
            return scroll_menu(items,8,20-menu_start_col);
        } else if (key==ESCAPE) {
            return -1;
        } else if (key==ENTER) {
            return selected;
        }
    }
}
```

Finalmente tenemos la función principal. En ella se usan todas las funciones que hemos escrito anteriormente para hacer que el programa funcione adecuadamente. También lee las teclas pulsadas con getch y si F1 o F2 son pulsadas, dibuja la ventana de menú correspondiente con draw_menu. Después llama a scroll_menu y deja que el usuario haga su selección desde los menús. Una vez tenga el valor devuelto por scroll_menu, borra las ventanas de menú e imprime el item seleccionado en la barra de mensajes.

Habría que mencionar ahora la función touchwin. Si se llama directamente a refresh sin llamar antes a touchwin después de cerrar los menús, el último menú abierto puede permanecer en pantalla. Esto es poque las funciones de menú no afectan para nada a stdscr y así, cuando se llama a refresh, éste no reescribe ningún caracter de stdscr ya que asume que dicha ventana no ha sido cambiada. touchwin pone todos los indicadores en la estructura WINDOW para decirle a refresh que todas las líneas de la ventana han cambiado y así en el próximo refresco de pantalla se reescribe la ventana completa incluso aunque sus contenidos no hayan cambiado. La información escrita en stdscr permanece ahí una vez que los menús se han cerrado porque los menús no escriben sobre stdscr, sino que lo hacen en las ventanas nuevas que han creado.

```
int main()
{
    int key;
    WINDOW *menubar,*messagebar;

    init_curses();

    bkgd(COLOR_PAIR(1));
    menubar=subwin(stdscr,1,80,0,0);
    messagebar=subwin(stdscr,1,79,23,1);
    draw_menubar(menubar);
    move(2,1);
    printw("Press F1 or F2 to open the menus. ");
    printw("ESC quits.");
    refresh();

   do {
        int selected_item;
        WINDOW **menu_items;
        key=getch();
        werase(messagebar);
        wrefresh(messagebar);
        if (key==KEY_F(1)) {
            menu_items=draw_menu(0);
            selected_item=scroll_menu(menu_items,8,0);
            delete_menu(menu_items,9);
            if (selected_item<0)
                wprintw(messagebar,"You haven't selected any item.");
            else
                wprintw(messagebar,
                  "You have selected menu item %d.",selected_item+1);
            touchwin(stdscr);
            refresh();
        } else if (key==KEY_F(2)) {
            menu_items=draw_menu(20);
            selected_item=scroll_menu(menu_items,8,20);
            delete_menu(menu_items,9);
            if (selected_item<0)
                wprintw(messagebar,"You haven't selected any item.");
            else
                wprintw(messagebar,
                  "You have selected menu item %d.",selected_item+1);
            touchwin(stdscr);
            refresh();
        }
    } while (key!=ESCAPE);

    delwin(menubar);
    delwin(messagebar);
    endwin();
    return 0;
}
```

Si copia el código en un fichero llamada example.c y elimina mis explicaciones puede compilarlo con

```
gcc -Wall example.c -o example -lcurses
```

y probar el programa. También puede descargar el código en el enlace que viene más abajo en la sección de referencias.

Conclusión

He explicado un poco el funcionamiento básico de las ncurses, lo suficiente para crear una buena interfaz para su programa. Aun así, las capacides de la librería no se limitan sólo a lo que he explicado aquí. Prodrá descubrir muchas más cosas en las páginas man a las que ya le he remitido con anterioridad, cuando las mire se dará cuenta de que la información presentada aquí es solamente una introducción.

Referencias

- El programa de ejemplo: example.c
- Página web de ncurses:www.gnu.org/software/ncurses/