Introduzione a Ncurses

by Reha K. Gerçeker
<gerceker/at/itu.edu.tr>

About the author:

Reha è uno studente di ingegneria informatica a Istanbul, in Turchia. Ama la libertà che Linux dà come piattaforma di sviluppo software. Passa la maggior parte del suo tempo davanti al computer, scrivendo programmi. Vorrebbe diventare un programmatore intelligente, un giorno.

Translated to English by:
Reha K. Gerçeker
<gerceker/at/itu.edu.tr>

Abstract:

Ncurses è una libreria che fornisce funzioni di mappatura dei tasti funzione, di disegno dello schermo e di gestione di finestre multiple non sovrapposte su testimali a caratteri.

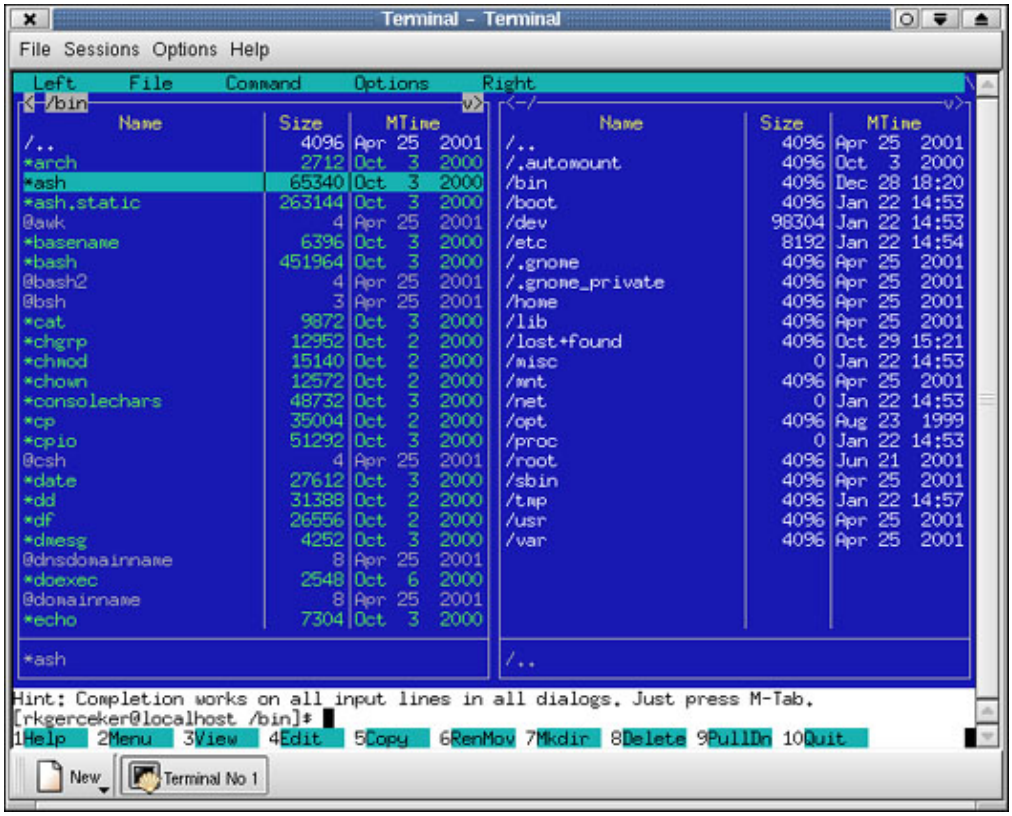
Cos'è ncurses?

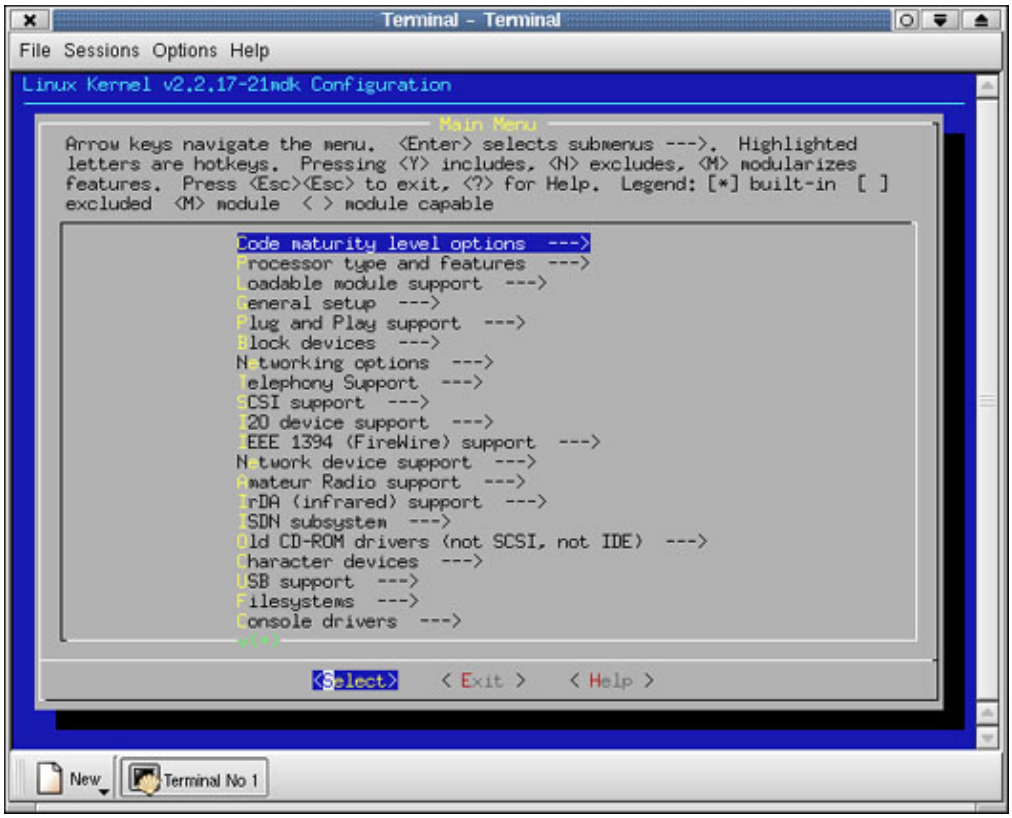
Volete che il vostro programma abbia un'interfaccia in modo testo colorata? Ncurses è la libreria che permette di avere finestre sui terminali a caratteri. Le cose che consente di fare sono:

- Usare l'intero schermo come si preferisce.
- Creare e gestire finestre.
- Usare 8 colori diversi.
- Aggiungere il supporto per il mouse ai programmi.
- Usare i tasti funzione della tastiera.

È possibile usare ncurses su qualsiasi sistema UNIX conforme allo standard ANSI/POSIX. Oltre a questo, la libreria è in grado di conoscere le capacità del terminale e di adeguarsi di conseguenza, fornendo un'interfaccia indipendente dal terminale. Per questo ci si può affidare a ncurses per progetti che dovrenno funzionare su diverse piattaforme e diversi terminali.

Midnight Commander è uno degli esempi scritti con ncurses. Anche l'interfaccia usata per la configurazione del kernel sulla console è scritta con ncurses. Potete vedere degli snapshot sotto.





Dove si scarica?

Ncurses viene sviluppata sotto GNU/Linux. Per scaricare l'ultima versione, avere informazioni dettagliate e trovare altri riferimenti, visitate www.gnu.org/software/ncurses/.

Le basi

Per usare la libreria dovete includere curses.h nel vostro codice sorgente e assicurarvi di fare un link alla libreria curses in compilazione. Per farlo dovete passare il parametro –lcurses a gcc.

Lavorando con ncurses è necessatio conoscere una struttura dati fondamentale. Si tratta della struttura WINDOW e, come si capisce facilmente dal nome, viene usata per rappresentare le finestre che create. Praticamente tutte le funzioni della libreria richiedono un puntatore a WINDOW tra i parametri.

I componenti più usati di ncurses sono le finestre. Anche se non create le vostre finestre, lo schermo è considerato una finestra. Così come il descrittore di FILE stdout della libreria di I/O standard rappresenta lo schermo (quando non ci sono redirezioni), ncurses ha un puntatore a WINDOW stdscr che copre lo stesso ruolo. Oltre a stdscr nelal libreria viene definito un altro puntatore a WINDOW chiamato curscr.Come stdscr rappresenta lo schermo, curscr rappresenta lo schermo attuale alla libreria. Potreste chiedere "Quale è la differenza?" Continuate a leggere.

Per usare le funzioni di ncurses nei vostri programmi dovrete richiamare la funzione initscr. Questa funzione alloca memoria per le variabili come stdscr, curscr e prepara la libreria all'uso. In altre parole tutte le funzioni di ncurses devono essere richiamate dopo initscr. Allo stesso modo dovrete richiamare endwin una volta terminato di usare ncurses. Questa funzione libera la memoria utilizzata da ncurses. Dopo aver usato endwin

non potrete usare le funzioni di ncurses finchè non richiamerete initscr.

Tra l'uso di initscr e quello di endwin assicuratevi di non mandare otput allo schermo con le funzioni della libreria di I/O standard. In caso contrario potreste ottenere un output su schermo confuso o corrotto. Quando ncurses è attiva usate le sue funzioni per mandare output verso lo schermo. Prima di chiamare initscr o dopo endwin potete fare quello che volete.

Ridisegno dello schermo: refresh

La struttura WINDOW non contiene solo l'altezza, la larghezza e la posizione della finestra, ma anche il contenuto della finestra. Quando scrivete su una finestra il suo contenuto cambia, ma questo non significa che il cambiamento appaia sullo schermo immediatamente. Per ottenere un ridisegno dello schermo dovete richiamare refresh o wrefresh

Questa è la differenza tra stdscr e curscr. Mentre curscr mantiene il contenuto dello schermo corrente, stdscr potrebbe contenere informazioni diverse dopo le chiamate alle funzioni di output di ncurses. Se volete trasferire su curscr i cambiamenti che avete fatto a stdscr dovete richiamare refresh. In altre parole, refresh è l'unica funzione che interagisce con curscr. È raccomandato di non trafficare con curscr e di lasciare che sia la funzione refresh ad aggiornarne il contenuto.

refresh ha un meccanismo per aggiornare lo schermo il più velocemente possibile. Quando la funzione viene chiamata cambia solo le linee modificate della finestra. Questo permette di risparmiare tempo CPU perchè evita di ristampare le stesse informazioni che sono già sullo schermo. Questo meccanismo è la ragione per cui le funzioni ncurses e quelle standard di I/O possono produrre pessimi risultati se usate insieme; quando le funzioni ncurses vengono chiamate, esse impostano un flag che informa refresh che la linea è cambiata, mentre questo non accade quando si chiama una funzione di I/O standard.

refresh e wrefresh fanno praticamente la stessa cosa. wrefresh prende come parametro un puntatore a WINDOW e aggiorna il contenuto di quella sola finestra. refresh() è equivalente a wrefresh(stdscr). Come dirò più avanti gran parte delle funzioni di ncurses hanno delle macro che applicano le stesse funzioni a stdscr.

Creare nuove finestre

Parliamo ora di subwin e newwin, le funzioni che permettono di creare nuove finestre. Entrambe queste funzioni prendono come parametri l'altezza, la larghezza e le coordinate dell'angolo in alto a sinistra della nuova finestra. Ritornano un puntatore a WINDOW che rappresenta la nuova finestra. Potete usare questo nuovo puntatore con wrefresh e altre funzioni di cui parlerò più avanti.

"Se fanno la stessa cosa, perchè duplicare le funzioni?" potreste chiedere. Avete ragione, sono un po' diverse. subwin crea la nuova finestra come sottofinestra di un'altra. Una finestra creata in questo modo eredita alcune proprietà della finestra genitore. Queste proprietà potrebbero essere cambiate in seguito senza influire sulla finestra genitore.

A parte questo c'è una cosa che lega le finestre genitore e figlie. L'array di caratteri che definisce il contenuto di una finestra viene condiviso tra la finestra genitore e la figlia. In altre parole i caratteri nell'intersezione tra le due finestre potrebbero essere cambiati da una qualsiasi delle due. Se il genitore scrive su tale rettangolo allora anche il contenuto della figlia cambia. È vero anche il contrario.

A differenza di subwin, newwin crea una finestra nuova di zecca. Questa finestra, a meno che non abbia figlie, non condivide il suo array di caratteri con nessun'altra. Il vantaggio di usare subwin è che la condivisione dell'array riduce l'uso di memoria. D'altra parte, quando le finestre iniziano a sovrapporsi a vicenda, l'uso di newwin ha i suoi vantaggi.

Potete creare le vostre sotto−finestre a qualsiasi profondità. Ogni sotto−finestra può avere proprie sotto−finestre, ma in questo caso ricordate che lo stesso array verrà condiviso da più di due finestre.

Quando avete finito con una finestra che avete creato potete cancellarla con la funziona delwin. Vi suggerisco di consultare la manpage per la lista di parametri di queste funzioni.

Scrivere sulle Finestre, Leggere dalle Finestre

Abbiamo parlato di stdscr, curscr, dell'aggiornamento dello schermo e della creazione di nuove finestre. Ma allora come scriviamo su una finestra? O come leggiamo dati da una finestra?

Le funzioni usate a questo scopo richiamano le loro controparti della libreria standard di I/O. Tra di esse ci sono printw al posto di printf, scanw al posto di scanf, addch al posto di putc o putchar, getch al posto di getc o getchar. Vengono usate come al solito, solo i nomi sono diversi. Allo stesso modo, addstr può essere usata per scrivere una stringa sulla finestra e getstr per leggere una stringa da una finestra. Tutte queste funzioni con una 'w' aggiunta all'inizio del nome e un puntatore a WINDOW come primo parametro eseguono il loro lavoro su una finestra diversa da stdscr. Per esempio printw(...) e wprintw(stdscr, ...) sono equivalenti, come refresh() e wrefresh(stdscr).

Sarebbe una lunga storia addentrarci nei dettagli di queste funzioni. Le pagine man sono il posto migliore dove imparare le loro descrizioni, i prototipi, i valori di ritorno e altre informazioni. Vi suggerisco di controllare le pagine man per ogni funzione che usate. Riportano informazioni dettagliate e preziose. L'ultima sezione di questo articolo, dove presento un programma di esempio, può servire anche come tutorial sull'uso delle funzioni.

Cursori Fisici e Logici

È necessario spiegare i cursori logici e fisici dopo aver parlato di leggere e scrivere sulle finestre. Quello che si intende per cursore fisico è il classico cursore lampeggiante sullo schermo e ce n'è solo uno. D'altra parte il cursore logico appartiene alle finestre ncurses e ogni finestra ne ha uno. Perciò ci possono essere più cursori logici.

Il cursore logico è il quadratino della finestra dove il processo di scrittura o lettura avrà inizio. Per questo, essere in grado di muovere il cursore logico significa poter scrivere in ogni posizione dello schermo o della finestra in qualsiasi momento. Questo è il vantaggio di ncurses rispetto alla libreria I/O standard.

La funzione che sposta il cursore logico è move o, come potete facilmente intuire, wmove. move è una macro per wmove, scritta per stdscr.

Un altro punto riguarda la coordinazione tra cursore fisico e logico. La posizione finale del cursore fisico sarà determinata dal flag _leave nella struttura WINDOW. Se _leave è impostato, il cursore logico viene spostato nella posizione del cursore fisico (nel punto in cui viene scritto l'ultimo carattere) dopo la fine della scrittura.

Se _leave non è settato il cursore fisico viene riportato alla posizione del cursore logico (dove viene scritto il primo carattere) dopo la scrittura. Il flag _leave è controllato dalla funzione leaveok.

La funzione che muove il cursore fisico è mvcur. A differenza delle altre, mvcur ha un effetto immediato e non attende il ridisegno. Se olete rendere invisibile il cursore fisico, potete usare la funzione curs_set. Controllate la pagina man per i dettagli.

Ci sono anche macro che raggruppano le funzioni di movimento e scrittura descritte sopra in un unico comando. Vengono descritte bene nelle stesse pagine man riguardanti addch, addstr, printw, getch, getstr, scanw, ecc.

Cancellare le Finestre

Abbiamo visto come scrivere sulle finestre. Ma come cancelliamo finestre, linee o caratteri>

Cancellare in ncurses significa riempire il carattere, la linea o il contenuto della finestra con spazi bianchi. Le funzioni che ho descritto sotto riempiono i caratteri necessari con spazi bianchi e quindi cancellano lo schermo.

Prima di tutto parliamo delle funzioni che cancellano un carattere o una linea. Le funzioni delch e wdelch cancellano il carattere che si trova sotto il cursore logico della finestra e spostano i caratteri che lo seguono sulla riga verso sinistra. deleteln e wdeleteln cancellano la riga su cui si trova il cursore logico e spostano in alto le linee seguenti.

Le funzioni clrtoeol e wclrtoeol cancellano tutti i caratteri sulla stessa linea alla destra del cursore logico. clrtobot e wclrtobot prima richiamano wclrtoeol per cancellare tutti i caratteri a destra del cursore logico, e quindi cancellano tutte le linee seguenti.

Oltre a queste esistono funzioni che cancellano l'intero schermo o la finestra. Ci sono due modi per cancellare tutto lo schermo. Il primo è di riempire tutti i caratteri con spazi bianchi e poi chiamare refresh, mentre il secondo è di usare il codice di controllo dello specifico terminale. Il primo metodo è più lento del secondo perchè richiede la riscrittura di tutti i caratteri sullo schermo mentre il secondo cancella tutto lo schermo immediatamente.

erase e werase riempiono l'array di caratteri di una finestro con spazi bianchi. Al prossimo refresh la finestra verrà cancellata. Se però la finestra occupa tutto lo schermo non è una buona idea usare queste funzioni, in quanto usano il primo metodo descritto sopra. Quando la finestra da cancellare ha le dimensioni dello schermo è meglio usare le funzioni descritte sotto.

Prima di addentrarci in altre funzioni è meglio menzionare il flag _clear. Esso è contenuto nella struttura WINDOW e se è settato chiede a refresh di mandare il codice di controllo al terminale quando viene richiamato. Al momento della chiamata refresh controlla se la finestra ha le dimensioni dello schermo (controllando il flag _FULLWIN) e in questo caso cancella lo schermo con l'apposito codice di controllo. Quindi scrive solo caratteri che non siano spazi bianchi sullo schermo. Questo rende la cancellazione dello schermo più veloce. Il motivo per cui il codice di controllo viene usato solo per finestre che riempiono tutto lo schermo è che esso cancella tutto lo schermo e non solo la finestra. Il flag _clear viene controllato dalla funzione clearok.

Le funzioni clear e wclear vengono usate per cancellare finestre delle dimensioni dello schermo. In effetti queste funzioni sono equivalenti a chiamare werase e clearok. Prima di tutto riempiono l'array di caratteri

della finestra con spazi bianchi. Quindi, impostando il flag _clear, cancellano lo schermo con il codice del terminale se la finestra è a tutto schermo oppure ridisegnano tutti i caratteri della finestra riempiendoli con spazi bianchi.

In definitiva, se sapete che la finestra è a tutto schermo usate clear o wclear. D'altra parte non c'è differenza tra l'uso di wclear o werase quando la finestra non è a tutto schermo.

Usare i Colori

I colori che vedete sullo schermo dovrebbero essere considerati come coppia di colori. Questo perché ogni carattere ha un colore di sfondo e uno di primo piano. Scrivere a colori con ncurses significa creare le proprie coppie di colori e usare queste coppie per scrivere su una finestra.

Proprio come initscr deve essere chiamata per inizializzare ncurses, start_color deve essere chiamata per inizializzare i colori. La funzione di cui avrete bisogno per creare le vostre coppie di colori è init_pair. Quando create una coppia con init_pair essa viene associata al numero che passate come primo parametro della funzione. A questo punto, quando vorrete usare questa coppia, vi riferirete ad essa chiamando COLOR_PAIR con il numero associato.

Oltre a creare le coppie di colori avrete bisogno delle funzioni necessarie a scrivere con diverse coppie. Questo viene fatto tramite le funzioni attron e wattron. Queste funzioni, fino alla chiamata di attroff o wattroff, fanno in modo che tutto quello che viene scritto sulla finestra corrispondente sia della coppia di colori che avete selezionato.

Ci sono anche le funzioni bkgd e wbkgd che cambiano la coppia di solori associata all'intera finestra. Quando vengono richiamate cambiano il colore di sfondo e di primo piano di tutti i caratteri della finestra. In questo modo, al prossimo refresh ogni carattere della finestra verrà riscritto con la nuova coppia di colori.

Nelle pagine man troverete i colori disponibili e i dettagli delle funzioni summenzionate.

Bordi attorno alle Finestre

Potete creare dei bordi attorno alle vostre finestre per dare un aspetto migliore al vostro programma. C'è una macro nella libreria chiamata box che lo fa per voi. A differenze di altre funzioni, non esiste una wbox; box accetta come argomento un puntatore a WINDOW.

Troverete i dettagli di box nella sua pagina man. C'è un'altra cosa che va menzionata. Mettere una finestra dentro un bordo significa semplicemente scrivere i caratteri necessari nell'array di caratteri della finestra che corrispondono ai suoi bordi. Se in futuro scriverete su tali caratteri il bordo ne sarà corrotto. Per prevenire questa situazione potete creare una finestra interna a quella originaria con subwin, applicate la box alla finestra originaria e usate la finestra interna per scrivere quando serve.

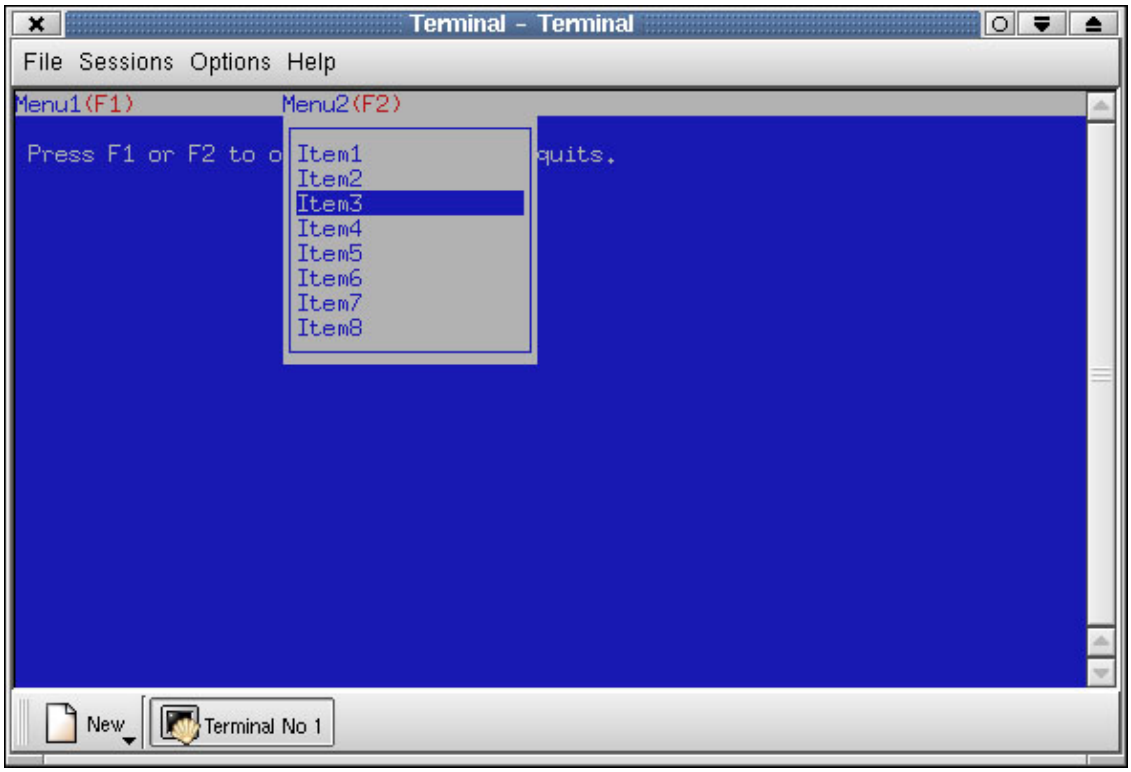
Tasti Funzione

Per poter usare i tasti funzione è necessario impostare il flag _use_keypad nella finestra da cui state cercando di leggere l'input. La funzione che imposta questo valore è keypad. Quando lo impostate potete ricevere input dalla tastiera normalmente con le funzioni di input.

In questo caso, se usate getch per ricevere i dati per esempio, dovete fare attenzione a salvare tali dati in un int piuttosto che in un char. Questo perché i valori numerici dei tasti funzione sono maggiori dei valori che una variabile di tipo char può contenere. Non avete bisogno di conoscere tali valori ma potete usare i nomi per loro definiti nella libreria. Tali valori sono elencati nella pagina man di getch.

Un Esempio

Analizzeremo ora un picolo programmino. In questo programma i menu vengono creati con ncurses e viene mostrato come selezionarne una voce. Un aspetto interessante di questo programma è l'uso delle finestre di ncurses per creare un effetto menu. Potete vedere uno snapshot qui sotto:



Il programma inizia con i soliti file di include. Quindi definiamo le costanti per i valori ASCII dei tasti enter ed esc.

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

#define ENTER 10
#define ESCAPE 27
```

La funzione qui sotto viene richiamata per prima alla partenza del programma. Chiama subito initscr per inizializzare ncurses e subito dopo chiama start_color per rendere possibile l'uso dei colori. Le coppier di

colori usate durante il programma verranno definite dopo. La funzione curs_set(0) rende invisibile il cursore fisico. noecho fa in modo che quello che si scrive da tastiera non venga riportato a schermo. Potete usarla anche per controllare cosa viene scritto e stampare a schermo solo le parti che volete visualizzare. La funzione echo dovrà essere richiamata quando necessario per disattivare questo effetto. La funzione qui sotto alla fine chiama keypad per abilitare i tasti funzione nell'input da stdscr. Questo si rende necessario in quanto useremo F1, F2 e i tasti cursore nel programma che segue.

```
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 funzione seguente crea una barra di menu che appare in alto sullo schermo. Controllando la funzione main sotto potete vedere che la barra di menu che appare come una singola linea in alto sullo schermo non è altro che una sottofinestra di una sola linea di stdscr. La funzione qui sotto riceve un puntatore a quella finestra come parametro, quindi prima ne cambia il colore di sfondo e poi scrive le voci di menu. Usiamo wstdadd per scrivere le voci di menu, ma potrebbe essere stata usata un'altra funzione. Fate attenzione alle chiamate a wattron, usate per scrivere con una diversa coppia di colori (numero 3) al posto della coppia di default (numero 2). Ricordate che la coppia numero 2 è stata impostata sulla prima linea da wbkgd. wattroff viene chiamata quando vogliamo tornare alla coppia di colori di default.

```
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 prossima funzione disegna i menu quando si preme F1 o F2. Per creare l'effetto dei menu viene creata una nuova finestra con lo stesso colore bianco della barra di menu, sopra la finestra blu che definisce lo sfondo. Non vogliamo che questa nuova finestra sovrascriva caratteri già scritti sullo sfondo. Essi dovrebbero tornare visibili dopo che il menu verrà chiuso. Per questo motivo la finestra di menu non può essere creata come sottofinestra di stdscr. Come potete vedere sotto la finestra items[0] viene creata con la funzione newwin e le altre 8 finestre per le sotto−voci vengono create come sottofinestre di items[0]. Qui items[0] viene usata per disegnare un bordo attorno al menu e le altre finestre di sotto−voci vengono usate per mostrare la sotto−voce selezionata nel menu e per non sovrascrivere i caratteri del bordo del menu attorno a loro. Per dare l'effetto della selezione di una sotto−voce è sufficiente cambiare il suo colore di sfondo rispetto alle altre.Questo viene fatto nella terzultima riga; lo sfondo della prima sotto−voce viene reso diverso dalle altre, in modo che quando il menu si apre, la prima sia selezionata.

```
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 prossima funzione si limita a cancellare la finestra di menu creata dalla funzione appena vista. Prima di tutto cancella le finestre delle sotto−voci con delwin, quindi libera la memoria allocata per i puntatori alle sotto−voci.

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

La funzione scroll_menu ci permette di spostarci tra e dentro i menu. Legge i tasti premuti sulla tastiera con getch. Se vengono premuti i tasti cursore−su o cursore−giù viene selezionata la voce precedente o seguente. Questo viene fatto cambiando il colore di sfondo della sotto−voce selezionata. Se vengono premuti i tasti cursore−sinistra o cursore−destra il menu aperto viene chiuso e viene aperto l'altro. Se viene premuto il tasto enter la sotto−voce selezionata viene ritornata. Se viene premuto ESC vengono chiusi i menu senza selezionare nulla. La funzione ignora altri tipi di input. In questa funzione getch è in grado di leggere i tasti funzione dalla tastiera. Ricordo che questo è possibile perchè la prima funziona, init_curses, richiamava keypad(stdscr, TRUE) e il valore di ritorno di getch viene memorizzato in un int invece che in un char, visto che il valore dei tasti funzione è maggiore di quello che può essere contenuto in un char.

```
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;
        }
    }
}
```

Per finire, c'è la funzione main. Essa usa tutte le funzioni scritte in precedenza e descritte sopra per far funzionare il programma. Legge anche i caratteri con getch e se viene premuto F1 o F2 disegna il menu corrispondente con draw_menu. Dopodiche chiama scroll_menu e lascia la possibilità all'utente di selezionare dal menu. Dopo il ritorno da scroll_menu cancella le finestre dei menu e scrive sulla barra messaggi la voce selezionata.

Dovrei menzionare la funzione touchwin. Se refresh fosse stato chiamato senza touchwin dopo che i menu sono stati chiusi, l'ultimo menu aperto sarebbe rimasti sullo schermo. Questo perché le funzioni per i menu non modificano stdscr e alla chiamata di refresh, esso non ridisegnerebbe stdscr poichè assumerebbe che la finestra non fosse sia cambiata. touchwin imposta tutti i flag della struttura WINDOW per informare refresh del cambiamento della finestra, cosicche al prossimo refresh la finestra venga completamente ridisegnata anche se il suo contenuto non è stato alterato. Le informazioni scritte su stdscr rimangono dopo che i menu sono stati chiusi poiché i menu non scrivono su stdscr ma piuttosto creano delle loro finestre.

```
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("Premi F1 o F2 per aprire i menu. ");
    printw("ESC esce.");
    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,"Non hai selezionato alcuna voce.");
            else
                wprintw(messagebar,
                  "Hai selezionato la voce %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,"Non hai selezionato alcuna voce.");
            else
                wprintw(messagebar,
                  "Hai selezionato la voce %d.",selected_item+1);
            touchwin(stdscr);
            refresh();
        }
    } while (key!=ESCAPE);

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

Se copiate il codice in un file chiamato example.c e togliete le spiegazioni, potete compilare il codice con

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

e provarlo. Potete anche scaricare il codice seguendo i link nel capitolo dei riferimenti.

Conclusioni

Ho parlato dei fondamenti di ncurses che possono essere sufficienti per creare una buona interfaccia per i vostri programmi. Le capacità della libreria però non sono limitate a quelle descritte qui. Scoprirete molte altre cose dalle pagine man che vi ho spesso chiesto di leggere e realizzerete che le informazioni che vi ho presentato qui sono solo una introduzione.

Riferimenti

- Il programma di esempio: example.c
- Il sito di ncurses:www.gnu.org/software/ncurses/