Appunti di sistemi

prof. Marcello Missiroli

Versione 0.1 - Maggio 2001

Indice

1 Interfaccia utente **7**

1.1 L'interfaccia utente . . . 8
1.1.1 Tipi di interfaccia utente . . . 8
1.2 L'interfaccia testuale . . . 9
1.2.1 Esempio . . . 9
1.3 L'interfaccia semigrafica (ncurses) -base . . . 10
1.3.1 Cos'è ncurses? . . . 10
1.3.2 Come si usa la libreria? . . . 10
1.3.3 Come si compila un programma con ncurses? . . . 11
1.3.4 Rilevazione di errori . . . 11
1.3.5 Schermo e cursore . . . 11
1.3.6 Il refresh . . . 12
1.3.7 Scrivere e leggere . . . 12
1.3.8 Un primo esempio: ruota dei venti . . . 13
1.3.9 Usare i Colori . . . 14
1.3.10 Un secondo esempio: fantasmagoria . . . 14
1.3.11 Cancellare . . . 15
1.4 Interfaccia semigrafica ncurses - avanzato . . . 15
1.4.1 Le finestre . . . 15
1.4.2 Funzioni relative alle finestre . . . 16
1.4.3 Bordi e linee . . . 16
1.4.4 Tasti Funzione . . . 17
1.4.5 Terzo esempio : printshow. . . . 17
1.4.6 Estensioni di ncurses . . . 18
1.4.7 Compatibilità conio-ncurses . . . 18
1.4.8 Esercizi . . . 18

1 Interfaccia utente

1.1 L'interfaccia utente

Agli albori dell'informatica, tutti gli utenti di computer erano essi stessi programmatori; non ci si si preoccupava molto degli utenti inesperti poiché questi, per definizione, non lo utilizzavano per niente. Nel corso degli anni, le cose sono profonamente cambiate e oggi il programmatore è costretto sempre a pensare in temini di utilizzo per utente finale: ogni programma deve esser chiaro, utilizzare comandi semplici e chiari, possbilmente essere predisposto per essere programmato per più paesi ("internationalization", o i18n) e più lingue ("localization", o l8n).

Tutto questo non fa altro che spostare la difficoltà dall'utente finale, che prima era costretto a impararsi comandi astrusi e diversi da programma a programma al programmatore che dovrà "pensare" di non conoscere per nulla il proprio software e cercare di immaginare come si comporterebbe un utente alle prime armi che - per inciso - ha pochissima voglia di leggere i manuali.

1.1.1 Tipi di interfaccia utente

Possiamo differenziare a quattro diversi tipi di interfaccia utente, al momento attuale

Testuale Questo tipo di programmi utilizzano come interfaccia, appunto i caratteri. L'unica struttura logica per l'interfaccia è la linea di testo. L'unica sorgente di input è la tastiera e l'unica di output è lo schermo. I vantaggi di questa interfaccia sono la estrema semplicità di programmazione, l'assoluta portabilità, bassissima occupazione di risorse (si memoriazza solo il carattere). Si tratta però di una interfaccia molto limitata e arida. Esempio: hello.c e tutti i programmi realizzati solo con le librerie `stdio` o `stream`

Semigrafica In questo modo è possibile utilizzare i caratteri ma con l'uso deli colore. L'iterfaccia è organizzata in schermate. E' possibile utilizzare risorse più raffinate quali i tasti freccia o il mouse, in aggiunta alla tastiera. Questo sistema rende la programmazione leggermente più complessa e meno portabile. L'occupazione di memoria resta comunque bassa. Esempio: `mc`

Grafica Al contrario dei casi precedenti, sarà possibile comandare il colore di ogni singolo pixel. Per poter scrivere sara' necessario che siano fornite funzioni che generino la configurazione di pixel necessaria per rappresentare ogni singolo carattere (eventualmente variando font e dimensioni). La programmazione è ovviamente più complessa, dipendente dall'hardware e dalla risoluzione grafica adottata, quindi assai poco portabile. L'occupazione di memoria diventa molto elevata (ogni pixel deve esplicitamente essere memorizzato, con un'occupazione direttamente proporzionale ai colori: 1 pixel in bianco e nero = 1 bit, ma un pixel con migliaia di colori = 2 bytes! Ne consegue che una schermata 800x600 a 24 bit richiede circa 1,4 Mbytes). Esempio: la maggior parte dei giochi in modalità SVGA.

A eventi In questo caso il programma non è un programma a se stante che comprende il sistema a finestra, il copia e incolla, la rete e altro ancora. Il programma deve fare parte di un ambiente integrato (Toolkit Windows, GTK+/Gnome, Qt/KDE, Fltk), ed occorre quindi conoscere le librerie utilizzate ed utilizzarle in modo estremamente preciso. Si hanno a disposizione tantissime funzionalità che occorre però implementare. L'esecuzione risulta spesso molto più lenta delle altre modalità. Esempio: OpenOffice.

In questa sede ci limiteremo a considerare i soli primi due casi, poiché i secondi richiedono una trattazione ben più ampia nonché conoscenze di programmazione già avanzate.

1.2 L'interfaccia testuale

Anche se questo tipo di interfaccia appare in declino, vi sono diversi casi in cui non è possibile fare altrimenti e si è costretti ad utilizzare questo strumento. In ogni caso è possibile fornire alcune indicazioni base quali:

† Presentare all'utente le scelte in modo chiaro, proponendo delle alternative.
Per esempio, potreste presentare un menu fatto in questo modo:
`$preferisci (C)ancellare, (R)ipetere o (U)scire:  ?`

† Prevedere soluzioni di default.
Proporre sempre una soluzione predefinita, che possa essere selezionata premento soltanto Invio. Per esempio:
`$Formattare il disco, (S)i o (N)o:  ?  [N]`

† Minimizzare l'input dell'utente
Quanto più l'utente deve scrivere, tanto più saranno possibili errori involontari.La soluzione ideale è richeidere un singolo carattere, ove possibile, magara accettando tanto la versione minuscola quanto maiuscola.

† Controllare l'input dell'utente
E' vostro compito controllare che l'utente non faccia errori: accettate solo input sicuro, controllando i valori numerici, la lunghezza delle stringhe e via dicendo.

1.2.1 Esempio

Questo semplice esempio mostra come gestire un semplice menu di scelta, con opzioni controllate.

```
#include <stdio.h>
int main(void){
char scelta;
printf(" Menu principale\n");
printf(" ===============\n\n");
printf("(M)odificare\n");
printf("(U)scire\n");
printf("(S)tampare\n");
printf("(C)ancellare\n");
printf("Cosa desideri fare? ");
scelta=toupper(getchar()); fflush(stdin);
if (scelta=='A')
printf("Hai scento di (M)odificare\n");
else if (scelta=='U')
     printf("Hai scelto di (U)scire\n");
     else if (scelta=='S')
```

```
            printf("Hai scelto di (S)tampare\n");
            else if (scelta=='C')
                 printf("Hai scelto di (C)ancellare;
                 else printf("La tua scelta '%c' e' errata\n", scelta);
return 0; }
```

1.3 L'interfaccia semigrafica (ncurses) -base

1.3.1 Cos'è ncurses?

Ncurses è la libreria che permette di

- † 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.
- † Funzionare su un grande numero di terminali diversi, anche attraverso la rete, fornendo un'interfaccia indipendente dal terminale.

Per scaricare l'ultima versione, avere informazioni dettagliate e trovare altri riferimenti, visitate [www.gnu.org/software/ncurses]. Se utilizzate Linux molto probabilmente avete Linux nella vostra distribuzione (provate a battere rpm -q ncurses), anche se forse vi mancano i file per sviluppare i programmi sotto ncurses (in questo caso installate il pacchetto ncurses-devel. Se siete sotto windows la vostra unica opzione è quella di installate CygWin.

1.3.2 Come si usa la libreria?

Supponendo di avere installato correttamente ncurses, la cosa migliore è utilizzare il seguente file-scheletro

```
# include <ncurses.h>
main(int argc, char **argv) {
/* eventuale codice di inizializzazione */
initscr();
/*
   vostro codice con ncurses
                                        */
endwin();
/* eventuale codice conclusivo          */
}
```

- † L'include è necessario per dire al compilatore che intendete utilizzare questa libreria. A seconda della distribuzione, potreste dover includere curses.h anziché ncurses.h - occorre fare qualche esperimento.

† initscr(); prepara la libreria per essere utilizzate: in altre parole le funzioni di ncurses possono essere utilizzate richiamate solo dopo una ciamata a initscr().

† Qui potete utilizzare liberamente il codice di ncurses (e tutto il codice che volete). **IMPORTANTE**! Dopo aver richiamato initscr() non è possbile utilizzare le normali funzioni di input/output quali printf() o cout!

† endwin()chiude l'utilizzo di ncurses. Se per caso volete riprendere l'utilizzo di ncurses partendo dalla situazione precedente dovete riavviare ncurses con l'istruzione refresh(), altrimenti reinizializzate tutto con initscr().

1.3.3 Come si compila un programma con ncurses?

In linea di massima come un programma normale, tranne per il fatto che occorre segnalare al compilatore che si intende utilizzare la libreria in questione. Il comando tipico diventa quindi

```
user@linux:~> gcc nomeprogramma.c -o nomeprogramma -lncurses
```

Per evitare di scrivere *ad infinitum* questa lunga e noiosa riga potete (1) lasciare aperto un terminale che utilizzerete a questo scopo (usando il tasto preccia in su) oppure (2) preparare un Makefile minimale contentene queste sole righe

```
all: nomeprogramma.c
            gcc nomeprogramma.c -o nomeprogramma -lncurses
```

facendo attenzione di utilizzare TAB per saltare lo spazio; in queso modo per compilare il programma basterà scrivere, nella directory opportuna,

```
user@linux:~> make
```

1.3.4 Rilevazione di errori

In linea di massima, tutte le funzioni di ncurses restituiscono un tipo intero. Se la funzione è andata a buon fine, il valore restituito è OK, altrimenti è ERR - fanno eccezione alcune funzioni che creano le finestre. Risulta così piuttsto semplice controllare se una funzione ha generato errori

1.3.5 Schermo e cursore

Una volta eseguito `initscr()`, lo schemo viene totalmente cancellato. Dal punto di vista di ncurses, lo schermo è una matrice di dimensione variabile, a seconda del computer, del sistema operativo e del programma che state utilizzando. L'angolo in alto a sinistra ha sempre le coordinate x=0 e y=0[1]. Dato che le dimensioni dello schermo sono variabili, ncurses possiede due variabili intere, `LINES` e `COLS` che contengono, rispettivamente le dimensioni, in caratteri, delle linee e delle colonne del terminale. L'angolo in basso a destra avrà quindi le coordinate x=COLS e y=LINES.

Il sistema prevede anche un cursore, che normalmente è invisibile, che indica la posizione dello schermo in cui si scriverà con la prossima istruzione. Dopo l'inizializzazione, tale punto è sempre (0,0), ma è facile spostarlo con varie istruzioni.

[1] nelle routine conio del dos le coordinate iniziano da 1.

int curs_set(int visibilità);

La funzione permette di stabilire se si vuole un cursore invisibile, visibile o *molto* visibile ponendo come parametri 0,1 o 2.

1.3.6 Il refresh

Uno dei concetti fondamentali di ncurses è quello di refresh.L'idea è che tutti i comandi di output che invierete **non sono visualizzati** immediatamente, ma sono memorizzate in una speciale zona di memoria. Per visualizzare effettivamente le cose occorre utilizzare l'istruzione `refresh()`. Questo tipo di approccio permette di velocizzare l'output, poiché è possibile preparare svariate modifiche e visualizzarle contemporaneamente.

I programmi con ncurses sono quindi costituiti da uno o più comandi di visualizzazione seguiti dall'istruzione `refresh()`.

1.3.7 Scrivere e leggere

La prima cosa che occorre imparare è come scrivere sullo schermo. Dato che non possiamo ricorrere alle funzioni standard, ncurses ci fornisce alcune funzioni particolari molto semplici.

int printw(char *fmt [, arg] ...);

Questa routine è la controparte di ncurses di printf, e richiede una stringa di formato (anche se non è necessario andare a capo a fine riga). La stampa viene effettuata a partire dalla posizione iniziale dello schermo. Esempio: `printw("Hello World!")` stampa la frase alla posizione corrente.

int scanw(char *fmt [, arg] ...);

..e questa è l'equivalente di scanf. Esempio: `scanw("%c",&i)` legge la variabile i alla posizione corrente.

int mvprintw(int y, int x, char *fmt [, arg] ...);

Simile alla precedente, ma permette di indicare le coordinate dello schermo. Notate che le coordinate orizzontali e verticale sono ***invertite*** rispetto alla loro posizione naturale. Esempio: `mvprintw(3,3,"La variabile vale %d", x)` stampa la frase indicata alla posizione 3,3.

int mvscanw(int y, int x, char *fmt [, arg] ...);

Per questa funzione potete attivare le vostre potenti abilità deduttive e capire che cosa fa!

int move(int y, int x);

Si limita a spostare il cursore alla posizione indicata. Esempio: `move( 10,10 );`

int getch(void); int mvgetch(int y, int x);

Questa routine funziona in modo assai simile alla `getchar()` di stdio, con la differenza che non servere premere INVIO; anzi, se un tasto è già stato premuto, la funzione ritorna immediatamente. In realtà si tratta di una funzione estrememente configurabile: è possibile avere l'eco sullo schermo del carattere premuto, oppure richiedere l'uso di invio. mvgetch permette di spostare il cursore prima dell'input.

int addch(char ch); int mvaddch(int y,int x);

Equivalente alla putchar di stdio. La seconda versione permette di spostare il cursore e stampare in un'unica operazione.

int noecho(void);

Una volta avviato, ncurses fa in modo che i caratteri battuti in fase di input siano visibili sullo schermo. Dopo aver richiamato `noecho()`, invece, quello che si scrive non appare sullo schermo. La funzione inversa è `echo()`.

int cbreak(void);

Con questa funzione la pressione di un tasto viene immediatamente resa disponibile al programma, senza attendere l'invio. Il suo inverso è `nocbreak();`

int timeout(int delay);

Questa importante funzione permette di regolare il comportamento del programma. Se si fornisce come parametro 0, `getch()` non aspetta che l'utente prema un tasto e prosegue, fornendo come valore di ritorno un errore (ERR). Se si fornisce un valore positivo n, il programma aspetta per n millisecondi e poi prosegue. Se si fornisce un valore negativo, il programma funziona in modo regolare, attendendo indefinitamente l'input dell'utente.

int napms(int ritardo);

Questa funzione si limita a bloccare il programma per un certo numero di millisecondi.

1.3.8 Un primo esempio: ruota dei venti

Questo semplice programma inizializza le librerie e stampa quattro parole ai quattro angoli del terminale. Se site in modalità grafica, provate a ridimesionare la finestra: noterete che le parole sono sempre corretamente posizionate (a patto di avere una finestra di almeno 2x13 caratteri).

```
#include <ncurses.h>
int main() {
initscr();
mvprintw(0, 0,"Nord-ovest"); mvprintw(LINES-1, 0,"Sud-ovest");
mvprintw(0, COLS-8,"Nord-est"); mvprintw(LINES-1, COLS-7,"Sud-est");
refresh();  getch();  endwin(); return 0; }
```

1.3.9 Usare i Colori

Dato che stiamo lavorando sullo schermo a caratteri, occorre tenere presente che in ncurses si intende una **coppia di colori**, piuttosto che un colore singolo: un colore rappresenta il colore del carattere, il secondo il colore dello sfondo. Per utilizzare i colori occorre

int start_color(void);

È obbligatorio richiamare questa funzione se si intendono utilizzare i colori. È buona norma richiamarla subito dopo `initscr()`.

int init_pair(short pair, short f, short b);

Questa funzione permette di creare le coppie di colori che poi saranno utilizzate per scrivere con i colori. Il primo parametro è un valore da 1 a 7 (il valore 0 è riservato al bianco su nero e non può essere - solitamente - modificato); il secondo è il colore del cattere, il terzo il colore dello sfondo.

Anche se è possbile utilizzare qualsiasi colore, definendone le componeti rosso, verde e blu con la funzione init_color, per la maggior parte degli scopi è più semplice utilizzare gli otto colori predefiniti che sono: COLOR_BLACK, COLOR_RED, COLOR_GREEN, COLOR_YELLOW, COLOR_BLUE, COLOR_MAGENTA, COLOR_CYAN e COLOR_WHITE. Per definire la coppia di colori bianco su blu occorrerà quindi usare un'istruzione come

```
init_pair(1,COLOR_WHITE,COLOR_BLUE);
```

int attron(int attibuti);

E' la funzione che dice quale coppia di colori utilizzare per le prossime istruzioni. All'interno è possibile specificare molitssimi attributi, ma quello che ci interessa è cambiare i colori con la macro `COLOR_PAIR(n)`.

bkgd(int attributi);

Cambia colori in un sol colpo all'intera finestra.

1.3.10 Un secondo esempio: fantasmagoria

Questo secondo programma mostra un semplice uso dei colori. Si noti, fra l'altro, che all'interno di `printw` il carattere '\n' mantiene il suo usuale significato di "a capo"; lo stesso vale per altri caratteri non stampabili

```
#include <ncurses.h>
int main() {
initscr(); start_color();
init_pair(1,COLOR_RED,COLOR_BLACK);
init_pair(2,COLOR_WHITE,COLOR_BLUE);
printw("Normale\n");
attron(COLOR_PAIR(1)); printw("Rosso su nero\n");
```

```
attron(COLOR_PAIR(2)); printw("Bianco su blu");
refresh(); getch();
printw("BOOOOM!"); bkgd(COLOR_PAIR(2));
refresh(); getch(); endwin(); return 0; }
```

1.3.11 Cancellare

Cancellare sembra facile, ma in realtà non lo è, perché abbiamo a disposizione tantissime possibilità.

int delch(void);

Cancella il carattere che si trova sotto il cursore e spostano i caratteri che lo seguono sulla riga verso sinistra. Inutile dire che esiste la versione mvdelch che permette di spostare il cursore e cancellare.

int deleteln(void);

Cancella la riga su cui si trova il cursore e sposta in alto le linee seguenti.

int insertln();

Inserisce una riga vuota alla posizione attuale del cursore, spingendo in basso le altre. In questo modo, di fatto, si cancella l'ultima riga in basso.

int clrtoeol(void);

Cancella tutti i caratteri sulla stessa linea alla destra del cursore

int erase(void);

Copia spazi bianchi in ogni posizione dello schermo, cancellandolo tutto. Si tratta in generale di una routine *lenta*, da utilizzare con circospezione.

1.4 Interfaccia semigrafica ncurses - avanzato

1.4.1 Le finestre

Come abbiamo visto, ncurses non lavora *direttamente* sulla memoria video ma su una copia in-memoria dei dati; la visualizzazione avviene solo esplicitamente quando si richiama la funzione refresh(). In realtà il meccanismo è molto più potente di quanto descritto finora e permette ulteriori livelli di flessibilità, quali i pannelli e le finestre di dialogo.

Finora abbiamo utilizzato solo le funzioni più semplici, e quindi abbiamo lasciato gestire tutto internamente a ncurses. Ma se volessimo un programma in cui l'output è presentato in diverse sezioni piuttosto indipendenti? Certo, possiamo gestire tutto tenendo conto delle posizioni delle righe e delle colonne, ma dopo qualche tentativo vi apparirà ovvio che il gioco non vale la candela: l'ideale sarebbe lasciare gestire tutto al calcolatore.

la struct WINDOWS

Ncurses lavora con una struttura fondamentale chiamata, con poca fantasia, WINDOW, che rappresenta la copia in memoria di una finestra. Tutte le funzioni ncurses fanno riferimento a una finestra; se ciò non pare subito evidente, è perché anche l'intero schermo è considerato una finestra. È però possibile creare una nuova finestra e quindi poter lavorare separatamente su questa nuova entità.

Per gestire questi dati occorre dichiarare una variabile strutturata di tipo WINDOWS in questo modo

```
WINDOWS *finestra
```

Notare che la variabile finestra è un *puntatore*, non una variabile. Dato che anche lo schermo intero è considerato una finestra, esiste una variabile predefinita di tipo window chiamata `stdscr` (da standard screen). In quewto modo potrete intervenire sui dati della finestra con alcune funzioni che vedremo tra poco. Per creare finestre nuove ci sono due funzioni subwin e newwin.

WINDOW *newwin (int nlinee, int ncols, int inizio_y, int inizio_x);

Questa funzione crea una nuova finestra alta un certo numero di linee, larga un certo numero di colonne, posizionata alla riga y e colonna x. Risulta completamente indipendente dalla finestra

int delwin (WINDOW *finestra);

Questa elimina dalla memoria la finestra, mettendo lo spazio di nuovo a disposizione del sistema. Va chiamata quando si ha finito di lavorare con una finestra.

1.4.2 Funzioni relative alle finestre

Ora dovremmo vedere come si opera sulle finestre, ma fortunatamente la struttura di ncurses è tale che... le conosciamo già. In pratica per ognuna delle funzioni che abbiamo visto ne esiste un'euivalente che funziona solo su una finestra. I nomi delle funzioni sono esattamente gli stessi delle funzioni già note, con una w aggiunta all'inizio del nome e la variabile della finestra come primo parametro. Per cui avremo `wprintw()`, `wscanw()`, `wrefresh()`, `wmvaddch()` ecc.

1.4.3 Bordi e linee

Potete creare dei bordi attorno alle vostre finestre per dare un aspetto migliore al vostro programma.

int box (WINDOW *finestra, char car_vert, char car_orizz);

Questa è una delle poche funzioni per la quale non è prevista la versione con la 'w' davanti, dato che richiede sempre il puntatore a una finestra. I caratteri car_vert e car_orizz sono i caratteri da utilizzare per creare i bordi. Potete semplicemente utilizzare un asterisco, ma la forma più diffusa ed elegante è la seguente, che usa i caratteri semigrafici disponibili.

```
box(finestra,ACS_VLINE,ACS_HLINE);
```

Altra cosa importante: nulla vi vieta di scrivere sui bordi, cancellandoli e distruggendo l'effetto grafico! State molto attenti o usate una finstra più piccola di quella originaria per scrivere.

1.4.4 Tasti Funzione

E' possibile utilizzare i tasti funzione con ncurses ma, come nel caso del colore, occore chiamare un'appostita funzione per l'attivazione.

int keypad(WINDOW *finestra, int vero_o_falso);

Questa funzione attiva o disattiva la gestione dei tasti funzione. Una volta attivato le funzioni come scanw ricevono i tasti funzione, ma perché la cosa funzioni occorre che i dati siano memorizzati in un int anziché un char, dato che i valori dei tasti funzione sono superiori ai vlaori standard ASCII. Non è, per fortuna, necessario impararsi il codice dei tasti freccia, poiché sono disponibili nel manuale di getch (`man getch`). In ogni caso, eccone alcuni tra i più significativi.

Freccia destra KEY_RIGHT

Freccia sinistra KEY_LEFT

Freccia in alto KEY_UP

Freccia in basso KEY_DOWN

Tasto funzione 1 KEY_F1

1.4.5 Terzo esempio : printshow.

Questo esempio dimostra come sia (relativamente) semplice utilizzare ncurses per realizzare semplici animazioni e dialoghi a comparsa.

```
#include <ncurses.h>
int main() {
int c,i,ciclo=TRUE;
WINDOW *dialogo;
initscr();
curs_set(0);
noecho();
timeout(500);
while(ciclo) {
     for (i=1; i<10 ; i++)
          { mvprintw(i,i,"Salve! Questo è un programma inutile! ");
            refresh(); c=getch(); if (c!=ERR) break; deleteln(); }
     if (c!=ERR) {
            dialogo= newwin (7,32,7,17);
            box(dialogo,'*','*'); timeout(0);
            mvwprintw(dialogo,2,3,"Vuoi davvero interrompere?");
```

```
            mvwprintw(dialogo,4,12,"(s)icuro?"); c=wgetch(dialogo);
            if (c=='s') ciclo=FALSE;
            delwin(dialogo); timeout(500); };
      erase(); refresh(); };
endwin();
return 0;
}
```

1.4.6 Estensioni di ncurses

Ncurses può offrire funzioni ancora più accurate, incluso l'interfacciamento con il mouse. In aggiunta esistono funzioni già pronte per l'uso dei dialoghi (*forms* library) e dei menù (*menu* library). Tali capacità esulano dalla portate di questa presentazione, ma possono facilmente essere provate consultando la documentazione di ncurses (solitamente posta sotto `/usr/doc` o `/usr/share/doc`).

1.4.7 Compatibilità conio-ncurses

Conio è una libreria di vasto utilizzo nel mondo windows e fornisce un sottoinsieme delle funzionalità di ncurses. In linea di massima è molto semplice trasporre un programma scritto per conio in ncurses e anche il contrario.

E' anche possibile, ma alquanto laborioso, scrivere programmi che possano essere compliati indifferentemente con conio e ncurses, anche se le librerie sono abbastanza simili per farlo. Un metodo pratico consiste nello scrivere un file chiamato conio.h sotto linux che ridefinisce i comandi conio con ncurses. Esempio di un possibile conio.h

```
#include <ncurses.h>
#define cputs(x) printw(x);refresh()
#define clrscr() erase();refresh()
#define gotoxy(x,y) move(y,x)
```

Questo approccio crea qualche problema per esempio per la gestione del colore, ma è praticabile per granparte dei programmi

1.4.8 Esercizi

1. Creare un piccolo albero di natale utilizzando gli asterischi e senza utilizzare ncurses.
2. Rifare l'esercizio 1 con ncurses.
3. Rifate l'esercizio 1 in modo che l'albero risulti sempre centrato nello schermo e proporzionato alla dimensione dello schermo
4. Create una piccola agenda che visualizzi, per ogni giorno, un massimo di 6 appuntamenti. Potete modificare gli appuntamenti.
5. Provate a creare un piccolo clone del gioco "Snake", ovvero un piccolo serpente che si muove sullo schermo e non deve toccare se stesso o i muri. La versione iniziale prevede un serpente lungo 4 caratteri, pilotabile tramite 4 tasti a vostra scelta. Potrete quindi migliorare la versione aggiungendo colori, usando i tasti freccia ecc.

Bibliografia

[1] Inrtroduzione a Ncurses di Reha K. Gerçeker <gerceker@itu.edu.tr>

[2] Ncurses tutorial, C. scene

[3] Ncurses tutorial, Italian Linux Documentation Project