Università degli Studi di Cagliari

Sistemi Operativi

Modulo di Laboratorio 11

01

Introduzione alla libreria ncurses

Grafica e interazione su terminale Linux

- **ncurses** (**n**ew **curses**) è una libreria di funzioni che consente di **gestire un'interfaccia utente testuale** su terminale Linux, emulando un'interfaccia grafica essenziale
  - È un'implementazione open source di una **precedente libreria curses** per sistemi Unix, alla quale sono state però introdotte ulteriori funzionalità
- Attraverso le **API messe a disposizione per il codice C** è possibile svolgere funzionalità grafiche su schermo o impiegare il mouse per interagire con il terminale
- Qualora il relativo pacchetto non fosse già installato nella propria distribuzione Linux, è possibile **installarlo con il seguente comando**

```
$  sudo apt install libncurses5-dev libncursesw5-dev
```

- In questo caso verrà installata la **versione 5** della libreria, a cui faremo riferimento

Inclusione e utilizzo di ncurses

- Per impiegare le API della libreria nel proprio codice C bisogna **includere l'opportuno file header** con le definizioni delle funzioni disponibili:

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

  - In diverse versioni, l'`include` potrebbe essere di `ncurses.h` invece che `curses.h`

- È inoltre necessario specificare l'impiego di tale libreria durante il processo di compilazione con `gcc`, aggiungendo l'**opzione `-l ncurses`**:

```
$  gcc <sorgente C> -l ncurses -o <nome output>
```

  - L'opzione **`-l`** esplicita la necessità di ricercare la libreria indicata nella fase di **l**inking
  - Spesso lo spazio è omesso, e l'opzione diventa `-l<libreria>` (`-lncurses`)

- Si noti inoltre che `curses.h` include già altri header come, per esempio, `stdio.h`, in quanto curses si appoggia alla libreria di I/O standard

Struttura base di un programma

- Un tipico main di un programma che fa uso della libreria ncurses è così strutturato:

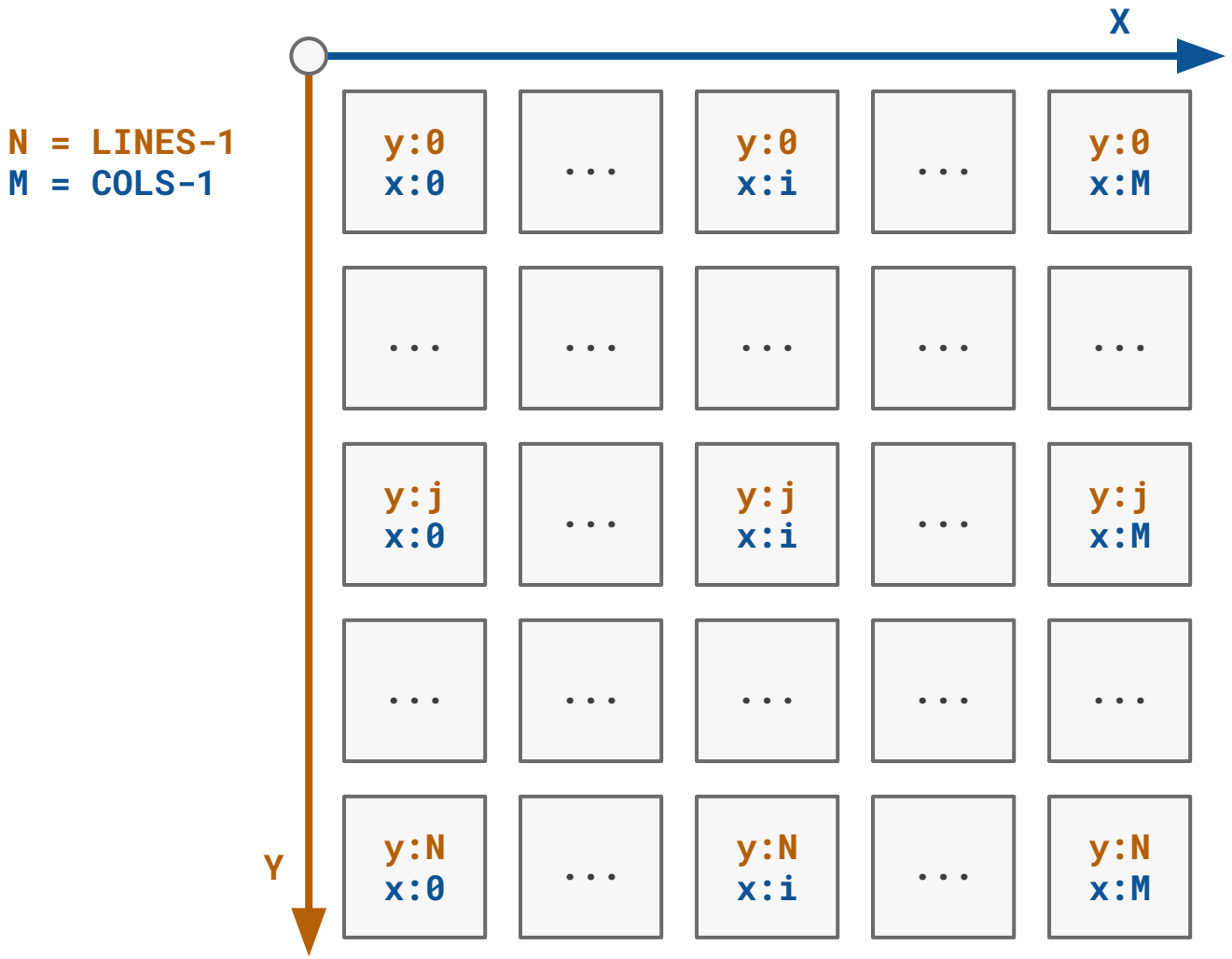
```c
#include <curses.h>
int main() {
    initscr();  // Inizializza ncurses e schermo
    //--- CODICE PROGRAMMA ---//
    endwin();   // Ripristina il normale utilizzo del terminale
}
```

  - **`initscr` inizializza la libreria**, permettendo solo dopo la sua chiamata di usare correttamente le funzioni di ncurses
    - Appena invocata cancella/pulisce lo schermo (l'area della shell/terminale)
    - **Rende indisponibili le funzioni di input/output standard** come, per esempio, `printf`
  - **`endwin` termina l'utilizzo della libreria** ncurses
    - Per riutilizzarla si invoca la funzione `refresh` o la si reinizializza con `initscr`

Funzionamento generale

- Tipicamente, tutte le funzioni di ncurses restituiscono un valore intero
  - Positivo (macro **`OK`**), per indicare che l'operazione si è conclusa con successo
  - Negativo (macro **`ERR`**), per indicare che si è verificato un errore
- In ambito ncurses si **gestisce lo schermo come una matrice di caratteri** identificati mediante coordinate (**`y`**, **`x`**), ovvero indice di riga e indice di colonna, infatti:
  - L'asse **`y`** è verticale e orientato verso il **basso**
  - L'asse **`x`** è orizzontale e orientato verso **destra**
  - L'angolo in **alto a sinistra** ha coordinate (**`0`**, **`0`**)
- La **dimensione dello schermo è recuperabile tramite `LINES`** e **`COLS`**, che indicano rispettivamente il numero di righe e colonne visibili nell'area di stampa
  - Pertanto, l'angolo remoto in basso a destra avrà coordinate (**`LINES-1`**, **`COLS-1`**)

Rappresentazione schermo e area di stampa



02

Input/Output testuale con ncurses

Cursore e visibilità

- In ogni istante il **cursore** è **posizionato in punto `(y,x)`** dello schermo
  - È la **posizione di default a cui si fa riferimento per scrivere/leggere** dallo schermo
  - Il cursore è inizialmente posizionato nel punto `(0,0)`
- La funzione **`move`** permette di collocare il cursore in un punto a piacere:

```c
int move(int y, int x);
```

- La funzione **`curs_set`** permette di impostare la visibilità del cursore:

```c
int curs_set(int visibility);
```

  - Il parametro `visibility` accetta i seguenti valori:
    - **`0`** nasconde il cursore, che risulta totalmente invisibile
    - **`1`** mostra il cursore staticamente
    - **`2`** mostra il cursore, facendolo lampeggiare per maggiore visibilità

Stampa su schermo

- Considerando che in ambito ncurses non sono disponibili le funzioni standard di input/output, occorre utilizzare le **funzioni di libreria alternative** per la stampa
- Sono fornite diverse funzioni per stampare sullo schermo testo e simboli, tra cui:
  - Stampa un **singolo carattere**

```c
int addch(const chtype ch);
```

  - Stampa una **stringa**, senza possibilità di formattazione

```c
int addstr(const char *str);
int addchstr(const chtype *chstr);
```

  - Stampa un **generico output**, grazie all'utilizzo degli **specificatori di formato** (versione equivalente della `printf`)

```c
int printw(const char *fmt, ...);
```

- La stampa inizia dal cursore, che avanza a destra con ogni carattere scritto

Lettura da schermo

- Oltre alla stampa di nuovo testo, è possibile **recuperare quanto già stampato**:
  - Legge un **singolo carattere**, restituendolo come valore di ritorno

```c
chtype inch(void);
```

  - Legge **tutto il testo fino alla fine della riga**

```c
int inchstr(chtype *chstr);
```

  - Legge il **testo fino ad n caratteri** o alla fine della riga

```c
int inchnstr(chtype *chstr, int n);
```

- La lettura parte dalla posizione corrente del cursore

Inserimento e cancellazione su schermo

- Sono infine possibili aggiornamenti contestuali dello schermo:
  - L'inserimento aggiunge del testo, facendo **avanzare il testo che segue**
  - La cancellazione rimuove del testo, facendo (nel caso) **retrocedere il testo che segue**
- Sono disponibili funzioni per inserire/cancellare caratteri, stringhe, righe …

Funzioni di inserimento
Inserisce un carattere, scorrendo a destra `int insch(chtype ch);`
Inserisce una stringa, scorrendo a destra `int insstr(const char *str);`
Inserisce una riga, scorrendo in basso `int insertln(void);`

Funzioni di cancellazione
Cancella un carattere, scorrendo a sinistra `int delch(void);`
Cancella tutto il testo che segue, fino a fine riga `int clrtoeol(void);`
Cancella una riga, scorrendo in alto `int deleteln(void);`
Cancella l'intero schermo `int erase(void);`

Refresh dello schermo

- Le operazioni di output che modificano lo stato dello schermo generalmente non restituiscono un feedback visivo immediato sul terminale
- Tutte queste operazioni scrivono su una **struttura dati intermedia che rappresenta il contenuto dello schermo**, ma la stampa effettiva sul terminale non è effettuata ad ogni suo aggiornamento, ed è invece occasionale
- Per **forzare la stampa**, richiedendo la trasposizione del contenuto della struttura dati intermedia sull'area di stampa del terminale, si ricorre ad una specifica funzione:

```c
int refresh(void);
```

- Altre funzioni (come `getch` che vedremo più avanti) potrebbero invocare la `refresh`
- Si deve **limitare il numero di chiamate a `refresh`**, essendo un'operazione onerosa

Acquisizione input

- Come per la stampa, anche per l'acquisizione bisogna ricorrere a funzioni alternative
- La libreria ncurses fornisce diverse funzioni per acquisire input dell'utente tramite la sua interazione con il terminale, tra cui:
  - Acquisisce un **singolo carattere**, restituendolo come valore di ritorno
    - Il tipo di ritorno intero è specificato per gestire gli eventuali codici di errore negativi

```c
int getch(void);
```

  - Acquisisce **una stringa**, senza possibilità di formattazione

```c
int getstr(char *str);
```

  - Acquisisce un **generico input**, accettando lo specificatore del formato di acquisizione e il puntatore a una variabile in cui memorizzarlo (versione equivalente della `scanf`)

```c
int scanw(char *fmt, ...);
```

Gestione avanzata del cursore

- Come ribadito più volte, ogni stampa o acquisizione fa riferimento alla posizione corrente del cursore, che deve essere spostato nella posizione di interesse
  - Lo spostamento del cursore prima di una operazione di I/O è così frequente che in ncurses **quasi ogni funzione possiede una variante che prende in input la posizione del cursore**, per effettuare lo spostamento in congiunta con l'operazione di interesse
  - Queste varianti sono caratterizzate dai caratteri **mv** anteposti al nome originale

```
1  int mvaddch(int y, int x, const chtype ch);
2  int mvgetch(int y, int x);
3  …
```

- Inoltre, dato che le operazioni di stampa possono spostare il cursore, si potrebbe essere interessati ad acquisire la sua posizione corrente con la seguente macro:

```
1  void getyx(WINDOW *win, int y, int x);
```

  - Si noti che non si passano i puntatori a y e x, poiché la macro se ne occupa per noi

Echoing dell'input

- Per comportamento predefinito, la libreria ncurses **visualizza sullo schermo i caratteri digitati** in fase di acquisizione dell'input (**echoing**)
- Si potrebbe tuttavia essere interessati ad acquisire un input per memorizzarlo in una variabile, senza volere che quanto digitato venga stampato anche sullo schermo
- È possibile attivare e disattivare l'echoing a piacimento:
  - **Attivazione** dell'echoing (già attivo di default)

```
1  int echo(void);
```

  - **Disattivazione** dell'echoing

```
1  int noecho(void);
```

Buffering dell'input

- Normalmente, durante l'acquisizione degli input, quello che viene digitato è **scritto in un buffer** e passato al programma solo dopo la pressione del tasto ENTER↵
  - Si parla in questo caso di **line buffering**
- È possibile inviare direttamente al programma quello che viene digitato disattivando il buffering con le opportune routine:
  - **Attivazione** del buffering

```c
int nocbreak(void);
```

  - **Disattivazione** del buffering

```c
int cbreak(void);
```

- Nell'interazione con l'utente tale modifica di comportamento è solo **percepibile con funzioni come `getch`**, in quanto funzioni che acquisiscono stringhe (come `scanw`) aspetteranno comunque la pressione del tasto ENTER↵ per segnalare la terminazione

Blocking dell'input

- Infine, le invocazioni alle funzioni di acquisizione dell'input sono di base bloccanti (**blocking**), ovvero l'**esecuzione si ferma sull'istruzione di acquisizione** finché non viene digitato l'input richiesto
- Anche questo comportamento può essere modificato, in due modi differenti:
  - Attivazione e disattivazione del comportamento bloccante

```
1  int nodelay(WINDOW *win, bool bf);
```

  - Attivazione e disattivazione di un timeout massimo (in millisecondi)
    - Impostando un timeout nullo si ottiene lo stesso effetto di `nodelay`

```
1  void timeout(int delay);
2  int notimeout(WINDOW *win, bool bf);
```

Esempio 01

- Acquisiamo un carattere e usiamolo per stampare una croce (X) a pieno schermo:

```c
#include <curses.h>
int main() {
    initscr();  // Inizializza ncurses e schermo
    curs_set(2);
    cbreak();

    mvaddstr(1, 1, "Inserisci un carattere a piacere:");
    char c = mvgetch(2,2);
    int size = ((LINES < COLS) ? LINES : COLS) - 2;
    mvprintw(3, 1, "Dimensione croce: %d\n Premi un tasto per stamparla ", size);
    getch();

    erase();
    for (int i=0; i<=size; i++) {
        mvaddch(i, i, c);
        mvaddch(i, size-i, c);
    }
    mvaddstr(LINES-1, 1, "Premi un tasto per terminare ");
    getch();

    endwin();   // Ripristina il normale utilizzo del terminale
}
```

03

Uso di colori con ncurses

Attivazione e definizione coppie colore

- In ncurses il colore per la stampa sullo schermo è definito da una coppia di colori:
  - Il **colore del carattere** da visualizzare
  - Il **colore del relativo sfondo**
- Per **attivare il supporto all'utilizzo dei colori** bisogna invocare la rispettiva routine
  - Ovviamente invocata dopo `initscr`, sui terminali che supportano i colori

```
1  int start_color(void);
```

- Per stampare con un colore bisogna prima definire esplicitamente una coppia colore

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

  - **`pair`** è un ID intero **da 1 a `COLOR_PAIRS-1`** (`0` è riservato a bianco su nero)
  - **`f`** è l'ID del colore per il carattere
  - **`b`** è l'ID del colore per lo sfondo

Definizione di colori

- Nel definire una coppia colore bisogna indicare gli ID dei **singoli colori**
- Sono disponibili diversi **colori predefiniti** i cui ID sono accessibili tramite macro:

Macro colore			
`COLOR_BLACK`	`COLOR_GREEN`	`COLOR_BLUE`	`COLOR_CYAN`
`COLOR_RED`	`COLOR_YELLOW`	`COLOR_MAGENTA`	`COLOR_WHITE`

- È altrimenti possibile **definire un colore** (o ridefinire) e **registrarlo permanentemente**

```
1  int init_color(short color, short r, short g, short b);
```

  - **`color`** è un ID intero **da `0` a `COLORS-1`**
    - Bisogna **stare attenti a non sovrascrivere il colore dietro gli ID predefiniti**
  - **`r`**, **`g`** e **`b`** sono le singole componenti RGB del colore, con **valore da `0` a `1000`**

Uso dei colori

- Per attivare e associare diversi attributi ai caratteri delle stampe a seguire, tra cui il loro colore, si ricorre alla funzione seguente:

```
1  int attron(int attrs);
```

- Per l'**attivazione di un colore** va usata in coppia alla macro per recuperare il colore:

```
1  attron(COLOR_PAIR(n));
```

- Si può poi interrompere ripetendo il comando con la funzione di disattivazione

```
1  int attroff(int attrs);
```

- Un'altra funzione utile può essere quella che permette di settare il colore di sfondo per tutta l'area di stampa

```
1  void bkgd(chtype ch);
```

Esempio 02

- Stampiamo del testo con diversi colori, e in chiusura cambiamo tutto lo sfondo

```c
#include <ncurses.h>
int main() {
    initscr();
    start_color();

    init_pair(1, COLOR_YELLOW, COLOR_BLUE);
    init_pair(2, COLOR_BLACK,  COLOR_WHITE);
    init_pair(3, COLOR_BLACK,  COLOR_RED);

    printw("Coppia di colori predefinita\n");

    attron(COLOR_PAIR(1)); printw("Coppia di colori giallo su blu\n");

    attron(COLOR_PAIR(2)); printw("Coppia di colori nero su bianco\n\n");

    attroff(COLOR_PAIR(2)); printw("Premi un tasto\n");
    getch();
    printw("Cambiato il colore dello sfondo in rosso... Premi un tasto per uscire");
    bkgd(COLOR_PAIR(3));
    getch();
    endwin();
}
```

04

Gestione e interazione avanzata

Finestre e standard screen

- Abbiamo anticipato come ncurses non scriva direttamente sul terminale, ma utilizzi una **struttura dati intermedia** in cui inserisce il contenuto da visualizzare a schermo
  - La struttura dati intermedia di astrazione dell'area di stampa è la **finestra** (**`WINDOW`**)
  - La stessa finestra è la base per altri strumenti più flessibili e avanzati, come i pannelli
- Tutte le funzioni su ncurses si **riferiscono sempre ad una finestra**
  - Quando inizializzata ncurses crea una finestra base: lo **standard screen** (**`stdscr`**)
  - Le funzioni che non richiedono esplicitamente una finestra usano lo standard screen
  - Lo standard screen occupa tutta l'area di stampa sul terminale

Creazione nuove finestre

- È possibile definire nuove finestre e gestire su ognuna di esse le operazioni di I/O in maniera separata
- Per **creare una nuova finestra** bisogna indicare:
  - Il numero di righe e colonne (se `0` di default va a pieno schermo)
  - Le coordinate dell'angolo superiore sinistro nel sistema di riferimento del terminale

```
1  WINDOW *newwin(int nlines, int ncols, int begin_y, int begin_x);
2  WINDOW *subwin(WINDOW *orig, int nlines, int ncols, int begin_y, int begin_x);
```

  - Usando `subwin`, la nuova finestra condivide la memoria della finestra originaria, così che fare il refresh dell'originale aggiorna anche la sotto-finestra
- **Dopo averla creata è possibile eliminarla**, deallocando le risorse impiegate

```
1  int delwin(WINDOW *win);
```

Interazione con le finestre

- È possibile svolgere ogni operazione di I/O indicando esplicitamente la finestra target
- **Quasi ogni funzione possiede la variante che prende in input la finestra** su cui agire
  - Queste varianti sono caratterizzate dal carattere `w` anteposto al nome originale
    - … ma eventualmente successivo ai caratteri `mv` per il posizionamento del cursore

```
1  int mvwaddch(WINDOW *win, int y, int x, const chtype ch);
2  int mvwgetch(WINDOW *win, int y, int x);
3  …
```

- Tra queste vi è anche `wrefresh`, per forzare la stampa del contenuto di una finestra
  - Notare che se si creano finestre sovrapposte tramite `newwin`, il contenuto visualizzato nell'area sovrapposta sarà quello della finestra su cui per ultima è invocato `wrefresh`

Delimitazione delle finestre

- Un insieme di funzioni molto interessante è quello che permette di **stampare e visualizzare chiaramente i bordi che delimitano una finestra**,
  - Indicando i caratteri da utilizzare per i bordi verticali e i bordi orizzontali

```c
int box(WINDOW *win, chtype verch, chtype horch);
```

  - Indicando i caratteri da utilizzare individualmente per i 4 bordi e i 4 angoli

```c
int wborder(WINDOW *win, chtype ls, chtype rs, chtype ts, chtype bs,
                         chtype tl, chtype tr, chtype bl, chtype br);
```

- È possibile utilizzare dei caratteri arbitrari, ma **solitamente si ricorre a caratteri semigrafici predefiniti** dati da `ACS_VLINE`, `ACS_HLINE` e altri

Esempio 03

- Creiamo e delimitiamo una nuova finestra

```c
#include <ncurses.h>
int main() {
    WINDOW *w1;

    initscr();
    curs_set(0);
    noecho();

    // Nuova finestra: 10 linee, 20 colonne, offset verticale di 1 e orizzontale di 2
    w1 = newwin(9, 20, 1, 2);

    // Aggiunge il bordo
    box(w1, ACS_VLINE, ACS_HLINE);

    // Stampa al centro della finestra
    mvwaddstr(w1, 4, 3, "Premi un tasto");

    // Attesa bloccante con refresh annesso
    wgetch(w1);

    endwin();
}
```

Interazione con i tasti funzione

- La libreria ncurses permette anche l'**utilizzo dei tasti funzione** per l'interazione
- La loro **attivazione** (o **disattivazione**) è gestita dalla routine seguente:

```
1  int keypad(WINDOW *win, bool bf);
```

- Una volta attivata, tutte le funzioni che ricevono un input (come `getch` o `scanw`) sono in grado di ricevere l'input anche dai tasti funzione
- Sono presenti anche delle **macro** che indicano i valori interi relativi ad alcuni tasti:
  - `KEY_DOWN` per la freccia verso il basso (valore intero `258`)
  - `KEY_UP` per la freccia verso l'alto (valore intero `259`)
  - `KEY_LEFT` per la freccia verso sinistra (valore intero `260`)
  - `KEY_RIGHT` per la freccia verso destra (valore intero `261`)
- Si può sempre far riferimento ai **valori interi espliciti** associati ad ogni tasto

Esempio 04

- Se si ha il dubbio su quale sia il codice intero associato ad un tasto è sempre possibile attivare i tasti funzione e premerli per acquisirli, stampando poi il valore restituito
- Il seguente programma svolge proprio questo compito:

```c
#include <ncurses.h>
int main() {
    int kcode;
    initscr();
    noecho();
    curs_set(0);
    keypad(stdscr, 1);
    mvprintw(1, 1, "Premi un tasto qualunque (q per uscire)");
    while(1) {
        kcode = getch();
        deleteln();
        mvprintw(1, 1, "Codice del tasto premuto (q per uscire): %d", kcode);
        if(kcode == (int)'q') break;
    }
    endwin();
}
```

Esempio 05

- Usiamo i tasti freccia per muovere un oggetto sullo schermo

```c
#include <stdlib.h>
#include <curses.h>
#define OBJECT '#'
int main() {
    initscr(); noecho(); curs_set(0); cbreak();
    keypad(stdscr, 1);
    box(stdscr, ACS_VLINE, ACS_HLINE);
    int x = COLS/2, y = LINES/2, c;  // Inizio al centro
    mvaddch(y, x, OBJECT); refresh();
    while(1) {
        c = (int)getch();
        mvaddch(y, x, ' ');
        switch(c) {
            case KEY_UP:    if(y > 1)         y -= 1; break;
            case KEY_DOWN:  if(y < LINES - 2) y += 1; break;
            case KEY_LEFT:  if(x > 1)         x -= 1; break;
            case KEY_RIGHT: if(x < COLS - 2)  x += 1; break;
            case (int)'q':  endwin(); exit(0);
        }
        mvaddch(y, x, OBJECT); refresh();
    }
}
```

Fine Modulo 11