

## Lezione 14

Scrivere la funzione `ladro` che gestisce la posizione del ladro sullo schermo del terminale. Fare in modo che l'entità dello spostamento del ladro corrisponda a quanto specificato dalla costante simbolica `PASSO`.

Si faccia in modo inoltre che il ladro non esca dall'area  $80 \times 24$  dello schermo durante i suoi spostamenti.

**Suggerimento:** si effettui una pausa tra uno spostamento e l'altro, per non rendere il movimento del ladro troppo "frenetico" sullo schermo (si utilizzi ad esempio la funzione `usleep`).

```
void ladro(int pipeout) {
    struct pos pos_ladro;
    long int r;
    int dx,dy;
    pos_ladro.x=1;
    pos_ladro.y=1;
    pos_ladro.c='$';

    write(pipeout,&pos_ladro,sizeof(pos_ladro));

    while(1) {
        r=random();

        if(r<RAND_MAX/2)
            dx=PASSO;
        else
            dx=-PASSO;

        if(pos_ladro.x+dx<1 || pos_ladro.x+dx>=MAXX)
            dx=-dx;

        pos_ladro.x+=dx;
        r=random();

        if(r<RAND_MAX/2)
            dy=PASSO;
        else
            dy=-PASSO;

        if(pos_ladro.y+dy<1 || pos_ladro.y+dy>=MAXY)
            dy=-dy;

        pos_ladro.y+=dy;
        write(pipeout,&pos_ladro,sizeof(pos_ladro));
        usleep(500000);
    }
}
```

Per determinare la nuova posizione del ladro, si calcola ad ogni iterazione un movimento casuale sia lungo l'asse x che lungo l'asse y. Siccome l'entità dello

spostamento deve essere di PASSO unità, il numero `r` casualmente generato dalla funzione `random` viene confrontato con `RAND_MAX/2` (l'intero massimo generabile da `random`). Se `r < RAND_MAX/2` lo spostamento è considerato positivo, altrimenti negativo.

I comandi condizionali

```
if(pos_ladro.x+dx<1 || pos_ladro.x+dx>=MAXX)
    dx=-dx;
```

e

```
if(pos_ladro.y+dy<1 || pos_ladro.y+dy>=MAXY)
    dy=-dy;
```

servono ad impedire che il carattere `$` che rappresenta il ladro esca dallo schermo (la direzione del movimento viene invertita nel caso vengano superati i bordi dello schermo del terminale).

Per completezza si riporta di seguito l'intero programma:

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

#define PASSO      10 /* entita' dello spostamento del ladro */

#define SU        65 /* Freccia su */
#define GIU       66 /* Freccia giu */
#define SINISTRA  68 /* Freccia sinsitra */
#define DESTRA    67 /* Freccia destra */

#define MAXX      80 /* Numero di colonne dello schermo */
#define MAXY     24 /* Numero di righe dello schermo */

/* Struttura per la comunicazione tra figli e padre */
struct pos {
    char c;          /* soggetto che invia il dato: ladro o guardia */
    int x;           /* coordinata x */
    int y;           /* coordinata y */
};

void ladro(int pipeout);
void guardia(int pipeout);
void controllo(int pipein);

int main() {
    int filedes[2];
    int pid_ladro;
    int pid_guardia;

    initscr(); /* inizializzazione dello schermo */
    noecho(); /* i caratteri corrispondenti ai tasti premuti non saranno
```

```

                * visualizzati sullo schermo del terminale
                */
curs_set(0); /* nasconde il cursore */

if(pipe(filedes)==-1) {
    perror("Errore nella creazione della pipe.");
    exit(1);
}

switch(pid_ladro=fork()) {
    case -1:
        perror("Errore nell'esecuzione della fork.");
        exit(1);
    case 0:
        close(filedes[0]);
        ladro(filedes[1]);
    default:

        switch(pid_guardia=fork()) {
            case -1:
                perror("Errore nell'esecuzione della fork.");
                exit(1);
            case 0:
                close(filedes[0]);
                guardia(filedes[1]);
            default:
                close(filedes[1]);
                controllo(filedes[0]);
        }
    }

kill(pid_ladro,1);
kill(pid_guardia,1);
endwin();

return 0;
}

void ladro(int pipeout) {
    struct pos pos_ladro;
    long int r;
    int dx,dy;
    pos_ladro.x=1;
    pos_ladro.y=1;
    pos_ladro.c='$';

    write(pipeout,&pos_ladro,sizeof(pos_ladro));

    while(1) {

```

```

r=random();

if(r<RAND_MAX/2)
    dx=PASSO;
else
    dx=-PASSO;

if(pos_ladro.x+dx<1 || pos_ladro.x+dx>=MAXX)
    dx=-dx;

pos_ladro.x+=dx;
r=random();

if(r<RAND_MAX/2)
    dy=PASSO;
else
    dy=-PASSO;

if(pos_ladro.y+dy<1 || pos_ladro.y+dy>=MAXY)
    dy=-dy;

pos_ladro.y+=dy;
write(pipeout,&pos_ladro,sizeof(pos_ladro));
usleep(500000);
}
}

void guardia(int pipeout) {
    struct pos pos_guardia;

    pos_guardia.c='#';
    pos_guardia.x=MAXX-1;
    pos_guardia.y=MAXY-1;

    write(pipeout,&pos_guardia,sizeof(pos_guardia));

    while(1) {
        char c;

        switch(c=getch()) {
            case SU:
                if(pos_guardia.y>0)
                    pos_guardia.y-=1;
                break;
            case GIU:
                if(pos_guardia.y<MAXY-1)
                    pos_guardia.y+=1;
                break;
            case SINISTRA:
                if(pos_guardia.x>0)

```

```

        pos_guardia.x--=1;
        break;
    case DESTRA:
        if(pos_guardia.x<MAXX-1)
            pos_guardia.x+=1;
        break;
    }

    write(pipeout,&pos_guardia,sizeof(pos_guardia));
}

}

void controllo (int pipein)
{
    struct pos ladro, guardia, valore_letto;
    ladro.x=-1;
    guardia.x=-1;

    do {
        read(pipein,&valore_letto,sizeof(valore_letto));

        if(valore_letto.c=='$') {

            if (ladro.x>=0) { /* cancello la 'vecchia' posizione del ladro */
                mvaddch(ladro.y,ladro.x,' ');
            }

            ladro=valore_letto;
        }
        else {

            if (guardia.x>=0) { /* cancello la 'vecchia' posizione della guardia */
                mvaddch(guardia.y,guardia.x,' ');
            }

            guardia=valore_letto;
        }

        mvaddch(valore_letto.y,valore_letto.x,valore_letto.c);
        curs_set(0);
        refresh();

    } while (guardia.x!=ladro.x || guardia.y!=ladro.y);
}

```