

## INSTALACIÓN DE LA LIBRERÍA NCURSES.

Personalmente, he realizado estas prácticas en Linux, por tanto para la instalación de la librería ncurses en este sistema operativo he ejecutado la siguiente orden:

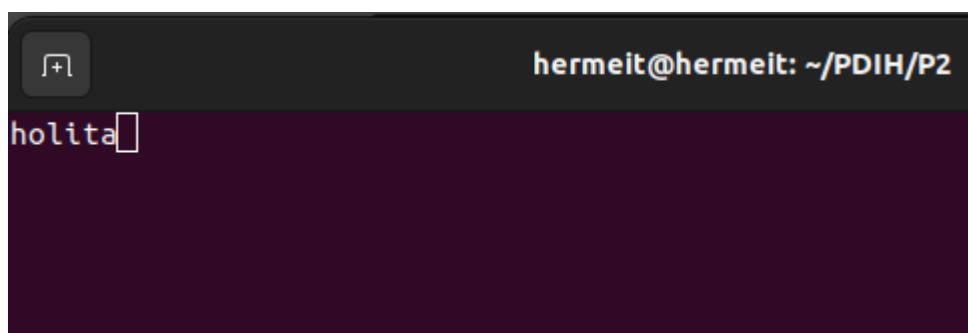
```
sudo apt-get install libncurses5-dev libncursesw5-dev
```

## EJECUTANDO ALGUNOS PROGRAMAS DE EJEMPLO

*hola.c*

```
1 #include <ncurses.h>
2
3 int main(){
4     initscr();
5     printw("holita");
6     refresh();
7     getch();
8     endwin();
9
10    return 0;
11 }
```

La función *initscr()* inicializa la pantalla y es obligatoria siempre, luego imprimimos la cadena “holita” en las coordenadas actuales (por defecto la 0,0), la función *refresh()* es para pasar el contenido de memoria intermedia a la pantalla real, esperamos una pulsación de tecla con la función *getch()* y acabamos con la función *endwin()* para liberar completamente la memoria. Esta sería su ejecución:



The screenshot shows a terminal window with a dark background. The title bar at the top reads "hermeit@hermeit: ~/PDIH/P2". Inside the terminal, the word "holita" is printed in white text, followed by a white cursor block. A small icon with a plus sign is visible in the top-left corner of the terminal window.

## ventana.c

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

int main(void){
    int rows, cols;

    initscr();

    if (has_colors() == FALSE){
        endwin();
        printf("El terminal no tiene soporte de color \n");
        exit(1);
    }

    start_color();
    init_pair(1, COLOR_YELLOW, COLOR_GREEN);
    init_pair(2, COLOR_BLACK, COLOR_WHITE);
    init_pair(3, COLOR_WHITE, COLOR_BLUE);
    clear();

    refresh();
    getmaxyx(stdscr, rows, cols);

    WINDOW *window = newwin(rows,cols,0,0);
    wbkgd(window, COLOR_PAIR(3));
    box(window, '|', '-');

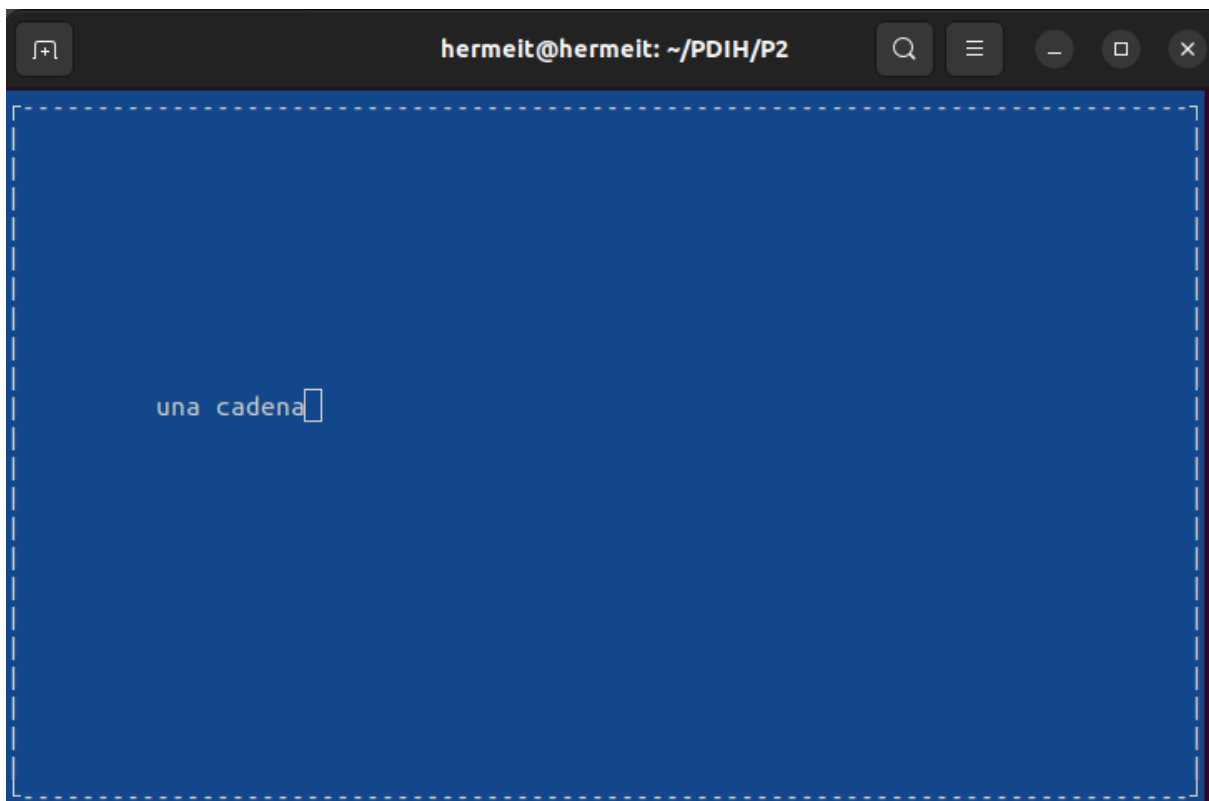
    mvwprintw(window, 10, 10, "una cadena");
    wrefresh(window);

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

Vamos a explicar brevemente lo que hace cada función:

- `init_pair()`: Crea pares de color de fondo y caracter y le asigna un número para luego poder usarla en las demás funciones.
- **`clear()`**: Limpia el contenido de la ventana.
- **`getmaxyx()`**: calcula el tamaño de número de filas y columnas en ese momento.

- ***newwin()*** y ***wbkgd()***: La primera crea una ventana del número de filas y columnas indicadas y la segunda establece un color de fondo y caracteres para esta.
- ***box()***: dibuja los marcos de la ventana recién creada.
- ***mvwprint()***: muestra cadenas de caracteres en la ventana especificada.
- ***wrefresh()***: refresca el contenido para mostrar en la ventana los últimos cambios realizados en ella.



## pelota.c

```
#include <unistd.h>
#include <ncurses.h>

#define DELAY 30000

int main(int argc, char *argv[]){
    int x=0, y=0;
    int max_y=50, max_x=50;
    int next_x=0;
    int direction=1;

    initscr();
    noecho();
    curs_set(FALSE);

    while(1){
        clear();
        mvprintw(y,x,"o");
        refresh();

        usleep(DELAY);

        next_x=x+direction;

        if(next_x >= max_x || next_x < 0){
            direction*=-1;
        }else{
            x+=direction;
        }
    }

    endwin();
}
```

Este programa mueve de izquierda a derecha el caracter 'o' por pantalla. Usamos la función **noecho()** que se encarga de que no haya eco por pantalla y la función **curs\_set()** que establece la apariencia del cursor.



## PROGRAMA ESTILO PONG

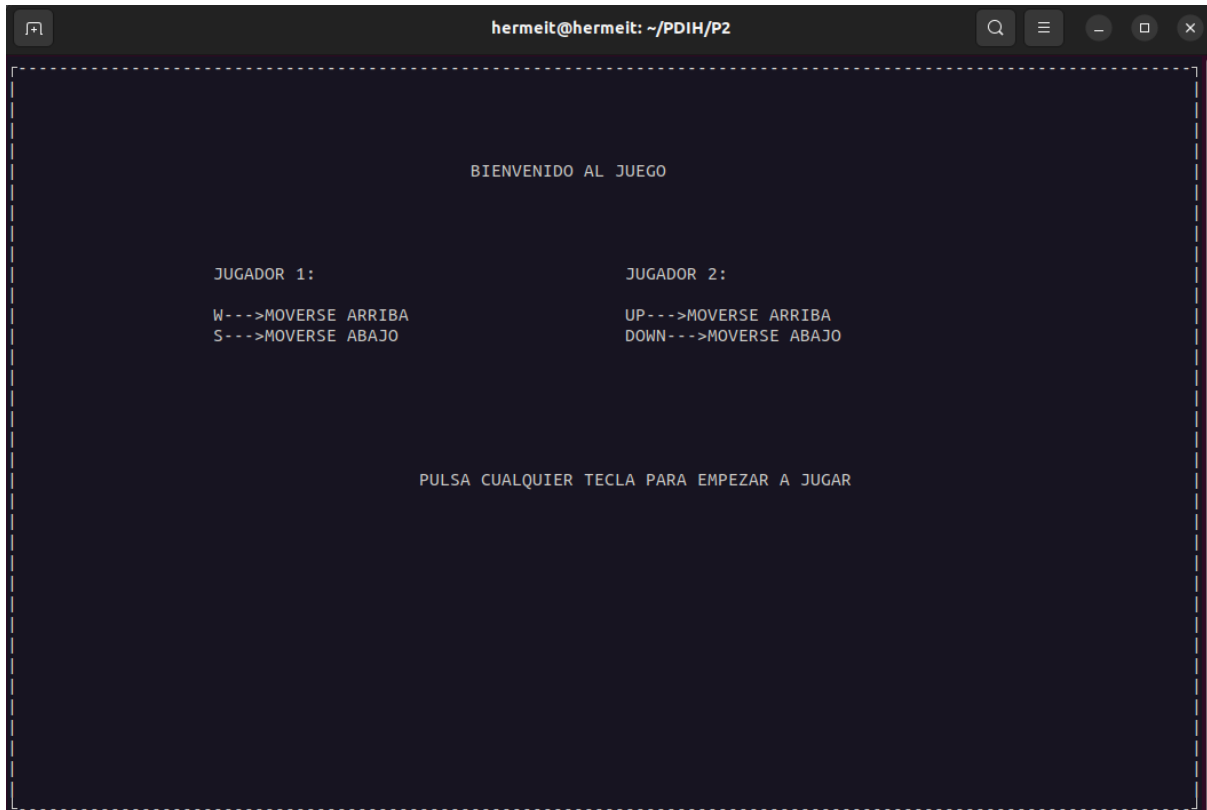
Para poner en práctica todos los conocimientos adquiridos en esta práctica, se pide implementar un programa estilo 'pong', el cual consta de dos jugadores que tienen que hacer rebotar una pelota en sus palas para pasársela al otro jugador, si el jugador 1 no consigue golpear la pelota, se suma un punto para el jugador 2 y viceversa, gana el primero que llegue a 5 puntos. El funcionamiento es el siguiente:

- Teclas *W* y *S*: serán las teclas de movimiento de la pala del jugador 1, *w* para moverla hacia arriba y *s* para moverla hacia abajo.
- Teclas *flecha arriba* y *flecha abajo*: Tendrán la misma funcionalidad que las anteriores solo que para el jugador 2.
- Tecla *P*: pausa la partida, volver a pulsar para reanudar.
- Tecla *escape*: Finaliza la partida.

Primero declaramos e inicializamos todas las variables que vamos a usar en nuestro juego.

```
1 #include <ncurses.h>
2 #include <unistd.h>
3 #include <stdlib.h>
4
5 #define DELAY 50000
6 #define PLAYER_PAIR 4
7 #define BALL_PAIR 3
8 #define NET_PAIR 5
9 #define PLAYER ' '
10 #define BALL 'O'
11
12 int main(int argc, char *argv[]){
13     int max_f, max_c, f_ball, c_ball, res_j1, res_j2;
14     int pala_j1_f, pala_j1_c;
15     int pala_j2_f, pala_j2_c;
16
17     int direccion_x = -1, direccion_y= -1, next_x= 0, next_y= 0;
18
19     int final_juego = 5;
20
21     initscr();
22
23     keypad(stdscr, true);
24     noecho();
25     curs_set(FALSE);
26
27     start_color();
28     init_pair(2, COLOR_WHITE, COLOR_BLACK);
29     init_pair(PLAYER_PAIR, COLOR_RED, COLOR_BLUE);
30     init_pair(BALL_PAIR, COLOR_BLACK, COLOR_RED);
31     init_pair(NET_PAIR, COLOR_BLACK, COLOR_GREEN);
32
33     clear();
34     refresh();
```

Al principio del juego se mostrará una pantalla con los controles básicos, esta pantalla esperará una pulsación de tecla para empezar el juego.



Para crear esta ventana se han implementado las siguientes líneas de código en las que creamos la ventana y mostramos por pantalla los datos con las funciones vistas en el ejemplo *ventana.c*

```
getmaxyx(stdscr, max_f, max_c);  
WINDOW *window = newwin(max_f,max_c, 0,0);  
wbkgd(window, COLOR_PAIR(2));  
box(window, '|', '-');  
  
mvwprintw(window, 5, 45, "BIENVENIDO AL JUEGO");  
  
mvwprintw(window, 10, 20, "JUGADOR 1:");  
mvwprintw(window, 12, 20, "W--->MOVERSE ARRIBA");  
mvwprintw(window, 13, 20, "S--->MOVERSE ABAJO");  
  
mvwprintw(window, 10, 60, "JUGADOR 2:");  
mvwprintw(window, 12, 60, "UP--->MOVERSE ARRIBA");  
mvwprintw(window, 13, 60, "DOWN--->MOVERSE ABAJO");  
  
mvwprintw(window, 20, 40, "PULSA CUALQUIER TECLA PARA EMPEZAR A JUGAR");  
  
wrefresh(window);  
  
getch();  
nodelay(stdscr,1);
```

Pasamos a la pantalla del juego, primero inicializamos los valores de las palas y de la pelota

```
//do{
f_ball = max_f/2;
c_ball = max_c/2;

res_j1=0;
res_j2=0;

pala_j1_c=1;
pala_j1_f=(max_f/2) -4;

pala_j2_c=max_c-1;
pala_j2_f=(max_f/2) -4;
```

Durante la ejecución del juego, el programa pintará constantemente mediante un bucle while las palas y la red, la pelota inicialmente estará en medio y mediante un if controlaremos su rebote, es decir si choca contra una pared o contra una pala, también actualizaremos los marcadores si algún jugador hace un punto. El bucle finalizará cuando uno de los marcadores llegue a 5.

```
while(res_j1 != final_juego && res_j2 != final_juego){

    clear();

    mvprintw(2, 35, "JUGADOR 1: %i", res_j1);
    mvprintw(2, 65, "JUGADOR 2: %i", res_j2);

    for(int i=1; i<max_f; i++){
        attron(COLOR_PAIR(NET_PAIR));
        mvaddch(i, max_c/2, PLAYER);
        attroff(COLOR_PAIR(NET_PAIR));
        move(i, max_c/2);
    }

    for(int i=0; i<8; i++){
        attron(COLOR_PAIR(PAYER_PAIR));
        mvaddch(pala_j1_f + i, pala_j1_c, PAYER);
        attroff(COLOR_PAIR(PAYER_PAIR));
        move(pala_j1_f + i, pala_j1_c);

        attron(COLOR_PAIR(PAYER_PAIR));
        mvaddch(pala_j2_f + i, pala_j2_c, PAYER);
        attroff(COLOR_PAIR(PAYER_PAIR));
        move(pala_j2_f + i, pala_j2_c);
    }

    attron(COLOR_PAIR(BALL_PAIR));
    mvaddch(f_ball, c_ball, BALL);
    attroff(COLOR_PAIR(BALL_PAIR));
    move(f_ball, c_ball);

    refresh();

    usleep(DELAY);
```

```

if(c_ball == pala_j2_c && (f_ball < pala_j2_f || f_ball > pala_j2_f +8)){
    res_j1++;

    f_ball = max_f/2;
    c_ball = max_c/2;
}

next_x = c_ball + direccion_x;
next_y = f_ball + direccion_y;

if(next_x >= max_c || next_x < 0){
    direccion_x *= -1;
}else{
    c_ball+=direccion_x;
}

if(next_y >= max_f || next_y < 0){
    direccion_y *= -1;
}else{
    f_ball+=direccion_y;
}

```

Por último disponemos de un switch que controla si en algún momento se ha pulsado una tecla, es decir controla el movimiento de las palas, pausar la partida o salir de esta si pulsamos escape

```

switch(getch())
{
case KEY_UP:
    if(pala_j2_f > 0){
        pala_j2_f--;
    }
    break;
case KEY_DOWN:
    if(pala_j2_f < max_f/2 +20){
        pala_j2_f++;
    }
    break;
case 'w':
    if(pala_j1_f > 0){
        pala_j1_f--;
    }
    break;
case 's':
    if(pala_j1_f < max_f/2 +20){
        pala_j1_f++;
    }
    break;
case 'p':
    getch();
    break;
case 0x1B:
    endwin();
    exit(0);
    break;
default:
    break;
}
}

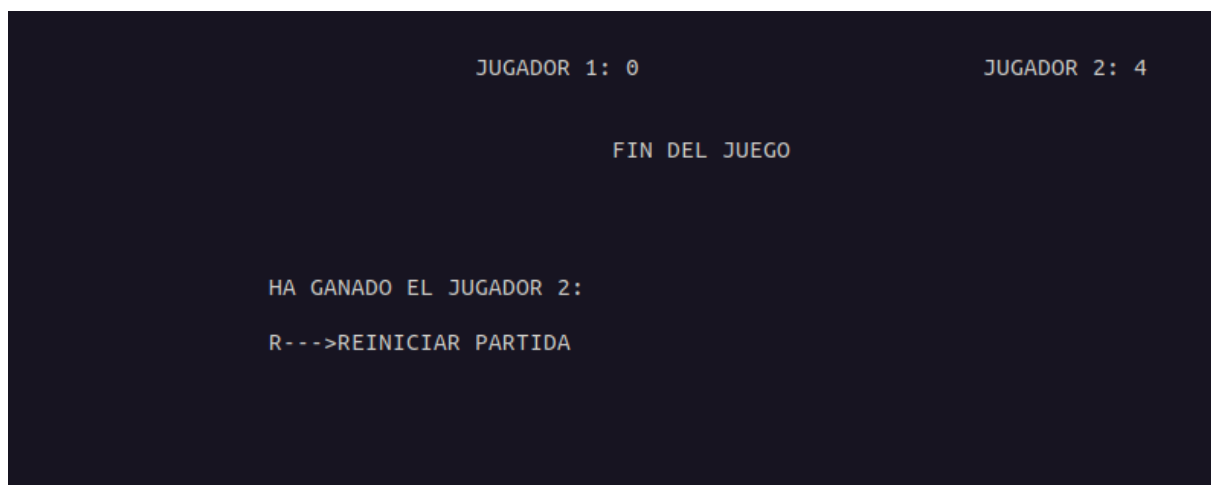
```



De esta forma la ejecución de nuestro juego quedaría así:



Cuando uno de los dos jugadores gane, se mostrará en pantalla el ganador y se esperará una lectura por teclado, si esta lectura es 'r' se reiniciará la partida y si no, finalizará el programa.



```

if(res_j1 == final_juego || res_j2 == final_juego){
    clear();

    if(res_j1 > res_j2){
        mvwprintw(window, 5, 45, "FIN DEL JUEGO");

        mvwprintw(window, 10, 20, "HA GANADO EL JUGADOR 1:");
        mvwprintw(window, 12, 20, "R--->REINICIAR PARTIDA");
    }else{
        mvwprintw(window, 5, 45, "FIN DEL JUEGO");

        mvwprintw(window, 10, 20, "HA GANADO EL JUGADOR 2:");
        mvwprintw(window, 12, 20, "R--->REINICIAR PARTIDA");
    }

    wrefresh(window);

    if(getchar() == 'r'){
        f_ball = max_f/2;
        c_ball = max_c/2;

        res_j1=0;
        res_j2=0;

        pala_j1_c=1;
        pala_j1_f=(max_f/2) -4;

        pala_j2_c=max_c-1;
        pala_j2_f=(max_f/2) -4;
    }else{
        endwin();
        exit(0);
    }
}

```