



ugr

Universidad  
de Granada

## PDIH

PERIFÉRICOS Y DISPOSITIVOS DE INTERFAZ HUMANA.

### Práctica 2: Uso de bibliotecas de programación de interfaces de usuario en modo texto.

---

**Autora:** Cristina María Crespo Arco

**Correo:** cmcrespo@correo.ugr.es

**Autor:** Andrés Piqueras Brück

**Correo:** andrespiqueras@correo.ugr.es

**Profesor:** Pedro A. Castillo Valdivieso



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

Curso 2021 - 2022

# Índice

<b>1. Instalación y configuración de ncurses</b>	<b>2</b>
1.1. Comprobar instalación. . . . .	2
1.1.1. hello.c . . . . .	2
1.1.2. ventana.c . . . . .	3
1.1.3. pelotita.c . . . . .	4
<b>2. Juego tipo “pong”.</b>	<b>6</b>
2.1. Código. . . . .	6
2.2. Ejecución pong.c. . . . .	10

# 1. Instalación y configuración de ncurses

Para instalar la librería ncurses en Ubuntu solo es necesario instalar los paquetes libncurses5-dev y libncursesw5-dev, para ello utilizaremos la siguiente orden:

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

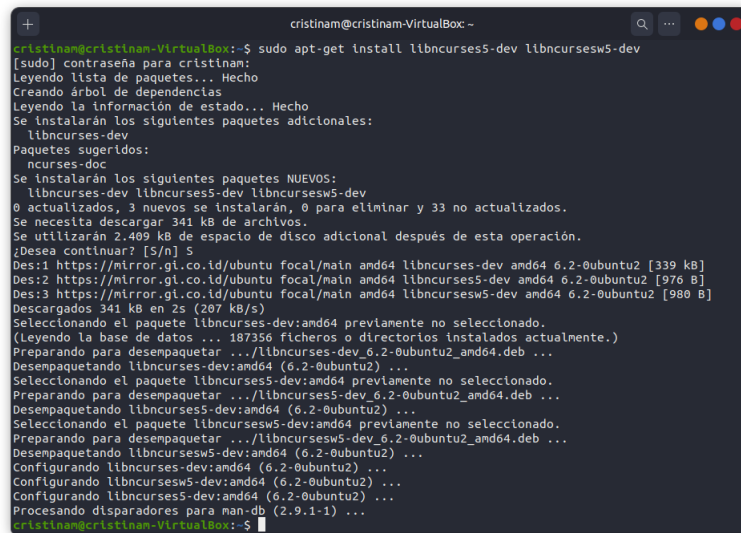
A screenshot of a terminal window titled 'cristinam@cristinam-VirtualBox ~'. The terminal shows the command 'sudo apt-get install libncurses5-dev libncursesw5-dev' being executed. The output includes the password prompt, package list reading, dependency tree creation, and state information reading. It lists the packages to be installed: libncurses-dev and libncursesw5-dev. It shows that 0 packages are updated, 3 new ones will be installed, and 0 will be removed. The total size of the packages is 341 kB. The user is prompted to continue, and they press 'Y'. The terminal then shows the download progress for three sources, the total download size of 341 kB, and the unpacking of the packages. Finally, it shows the configuration of the packages and the processing of man-db.

Figura 1: Instalación de la librería ncurses.

## 1.1. Comprobar instalación.

Para comprobar que se ha instalado correctamente la librería ncurses hemos ejecutado los programas de ejemplo que encontramos en el documento de esta práctica.

### 1.1.1. hello.c

- Código:

```
#include <ncurses.h>
#include <stdio.h>
int main() {
    initscr();
    printf("Hello World!");
    refresh();
    getch();
    endwin();
    return 0;
}
```

- Ejecución:

```
$gcc hello.c -o hello -lncurses && ./hello
```

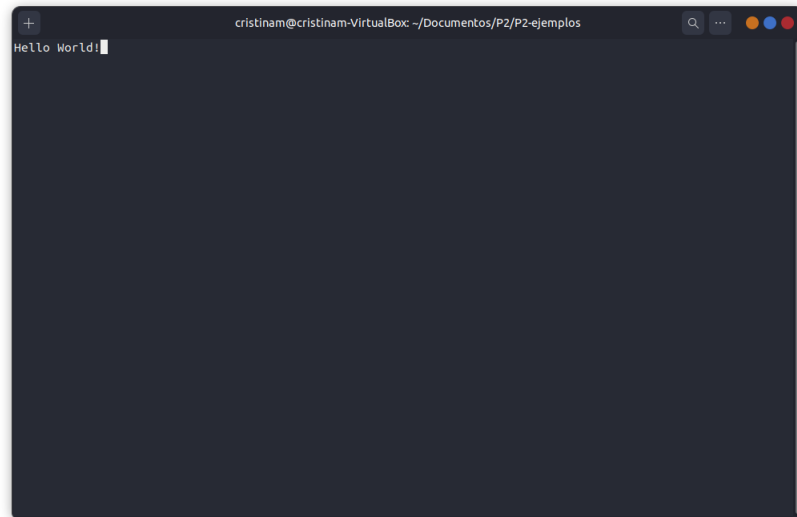


Figura 2: Ejecución hello.c.

### 1.1.2. ventana.c

- Código:

```
#include <stdlib.h>
#include <ncurses.h>
int main(void) {
    int rows, cols;
    initscr();
    if (has_colors() == FALSE) {
        endwin();
        printf("Your terminal does not support 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;
}

```

- Ejecución:

```
$gcc ventana.c -o ventana -lncurses && ./ventana
```

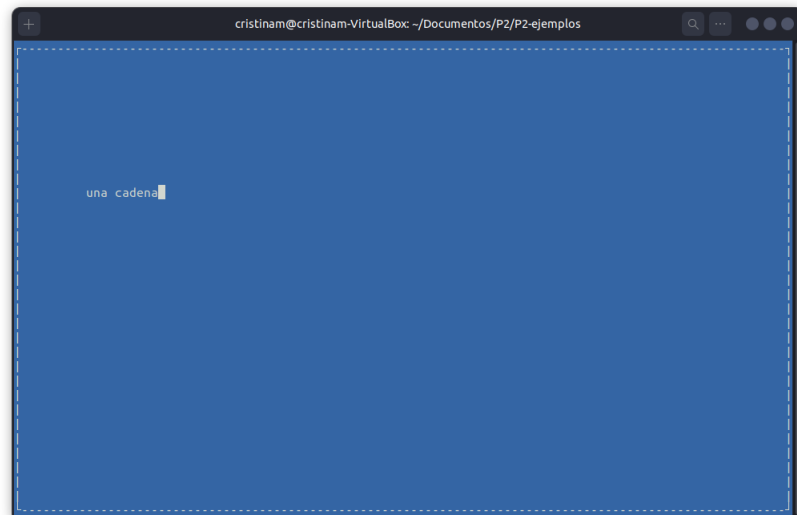


Figura 3: Ejecución ventana.c.

### 1.1.3. pelotita.c

- Código:

```

#include <ncurses.h>
#include <unistd.h>
#define DELAY 30000
int main(int argc, char *argv[]) {
    int x = 0, y = 0;
    int max_y = 20, max_x = 20;
    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();
}

```

■ Ejecución:

```
$gcc pelotita.c -o pelotita -lcurses && ./pelotita
```

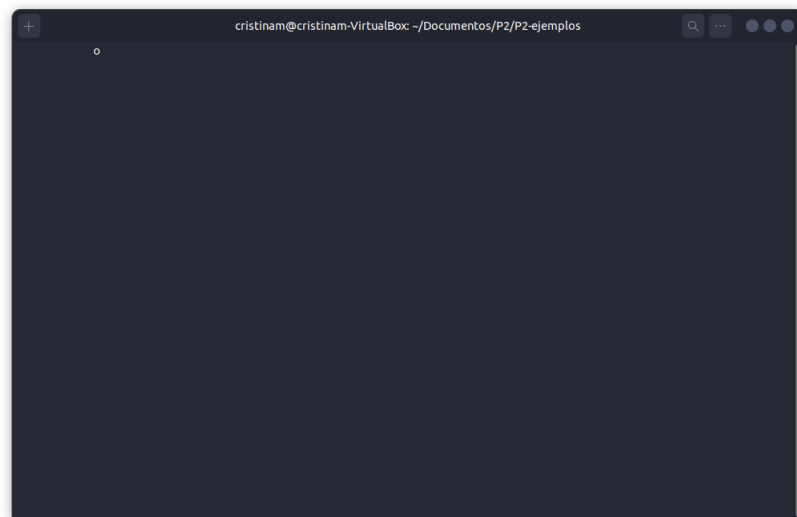


Figura 4: Ejecución pelotita.c.

## 2. Juego tipo “pong”.

<https://github.com/cmcrespo00/PDIH/blob/main/P2/pong.c>

### 2.1. Código.

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

#define DELAY 5000
typedef struct{short int x, y, puntos; bool mover_horizontal,
               mover_vertical;} objeto;
```

Hemos creado un Struct para almacenar los datos de las barras y la pelota. Los datos que vamos a necesitar son las coordenadas 'x', 'y', los puntos que llevan acumulados cada barra y luego, dos boolean para comprobar la dirección en la que se mueve la pelota.

```
int main() {
    objeto pantalla;
    int i = 0, cont=0;
    bool terminar = false;

    initscr();
```

A continuación, hemos creado cuatro pares de colores que usaremos para la puntuación, para la línea que separa los dos campos, la pelota y las barras de cada jugador, respectivamente.

```
start_color();
init_pair(1,COLOR_MAGENTA,COLOR_BLACK);
init_pair(2,COLOR_MAGENTA,COLOR_MAGENTA);
init_pair(3,COLOR_YELLOW,COLOR_BLACK);
init_pair(4,COLOR_YELLOW,COLOR_YELLOW);

 keypad(stdscr, true);
 noecho();
 curs_set(FALSE);
```

Hemos usado la función `getmaxyx(stdscr, pantalla.y, pantalla.x)` para calcular el tamaño de la terminal y así adaptar nuestro juego a ese tamaño. Esto se hace más adelante al crear los objetos que vamos a utilizar en nuestro programa (la pelota y las barras).

```
getmaxyx(stdscr, pantalla.y, pantalla.x);
```

```

objeto barra_izquierda = {1, pantalla.y/2, 0, false, false},
    barra_derecha = {pantalla.x-2, pantalla.y/2, 0, false, false},
    pelota = {pantalla.x/2, pantalla.y/2, 0, false, false};

```

Seguidamente encontramos la implementación de la página principal, donde podemos observar el título y justo debajo las instrucciones para poder jugar.

```

mvprintw(1,0,
"          *****          ***          \n"
"          ***      *****      ***      *****          \n"
"          ***      ***      ***      ***      ***      ***          \n"
"          ***      *****      ***      *****          \n"
"          ***      ***          ***      ***          \n"
"          ***      *****      ****      *****          \n"
"          *****          \n"
"          ***      ***      *****      *****      *****          \n"
"          ***      ***      ***      ***      ***      ***      ***      ***          \n"
"          *****      ***      ***      ***      ***      ***      ***          \n"
"          ***          ***      ***      ***      ***      *****          \n"
"          ***          *****      ***      ***          ***          \n"
"          *****          \n"
"*****\n"
"*Instrucciones para poder jugar:          *\n"
"* Para comenzar el juego pulsa cualquier tecla.          *\n"
"* Jugador de la derecha: Controles 'flecha arriba' y 'flecha abajo'.          *\n"
"* Jugador de la izquierda: Controles 'w' y 's'.          *\n"
"* Para pausar el juego pulse 'p'.          *\n"
"* Para salir pulse ESC.          *\n"
"*****\n");

```

Gracias al siguiente bucle for el programa se ejecuta indefinidamente hasta que se pulse la tecla ESC. A continuación, con los condicionales siguientes se comprueba que la pelota cambia de dirección al chocar contra los bordes de la pantalla.

```

getch();
for (nodelay(stdscr,1); !terminar; usleep(DELAY)) {
    if (++cont % 16 == 0){
        if ((pelota.y == pantalla.y-1) || (pelota.y == 1))
            pelota.mover_vertical =! pelota.mover_vertical;
        if ((pelota.x >= pantalla.x-2) || (pelota.x <= 2)){
            pelota.mover_horizontal =! pelota.mover_horizontal;

```

Con el condicional de abajo, comprobamos si la pelota ha chocado con las barras o, por el contrario, no ha dado en la barra, es decir, algún jugador ha perdido. El siguiente condicional comprobamos que jugador ha perdido y añade los puntos al jugador contrario. Después, mueve la pelota al centro del juego para jugar el próximo punto.



```

if ((pelota.y == barra_derecha.y-1) ||
    (pelota.y == barra_izquierda.y-1)) {
    pelota.mover_vertical = false;
} else if ((pelota.y == barra_derecha.y+1) ||
    (pelota.y == barra_izquierda.y+1)) {
    pelota.mover_vertical = true;
} else if ((pelota.y != barra_derecha.y) &&
    (pelota.y != barra_izquierda.y)){
    if (pelota.x >= pantalla.x-2)
        barra_derecha.puntos++;
    else
        barra_izquierda.puntos++;
    pelota.x = pantalla.x/2;
    pelota.y = pantalla.y/2;
}
}

```

Comprobamos el estado de los boolean mover\_ correspondientes a la pelota y según el estado se mueve el objeto pelota en una dirección o en la otra.

```

if(pelota.mover_horizontal)
    pelota.x = pelota.x+1;
else
    pelota.x = pelota.x-1;
if(pelota.mover_vertical)
    pelota.y = pelota.y+1;
else
    pelota.y = pelota.y-1;

```

Para evitar que desaparezcan las barras cuando mantenemos pulsado el botón de subir o el de bajar, hemos indicado que cuando las barras se vayan a salir de la vista esta aparezca por la dirección contraria, es decir, si sale por arriba la barra esta aparecerá por debajo y viceversa.

```

if (barra_derecha.y <= 1)
    barra_derecha.y = pantalla.y-2;
if (barra_derecha.y >= pantalla.y-1)
    barra_derecha.y = 2;

if (barra_izquierda.y <= 1)
    barra_izquierda.y = pantalla.y-2;
if (barra_izquierda.y >= pantalla.y-1)
    barra_izquierda.y = 2;
}

```

Hemos implementado el siguiente "switch" para indicar los controles que se van a utilizar para jugar. Además, hemos incluido el botón ESC para salir del juego y pulsando la letra 'p' se pausa el juego, para volver a activar el juego es necesario pulsar cualquier otra tecla.

```
switch (getch()) {
    case 'w':      barra_izquierda.y--; break;
    case 's':      barra_izquierda.y++; break;
    case KEY_DOWN: barra_derecha.y++; break;
    case KEY_UP:   barra_derecha.y--; break;
    case 'p':      getch(); break;
    case 0x1B:     endwin(); terminar++; break;
}
```

Por último, como hemos mencionado anteriormente, hemos creado diferentes pares de colores y, ahora, vamos a aplicar cada par al objeto correspondiente, mediante la función `attron(COLOR_PAIR(numParColor))`, donde `numParColor` es el número del par de color creado al inicio del programa.

```
erase();
attron(COLOR_PAIR(1));
mvprintw(0, pantalla.x/2-3, "%i      %i", barra_derecha.puntos,
        barra_izquierda.puntos);
attroff(COLOR_PAIR(1));

attron(COLOR_PAIR(2));
mvvline(0, pantalla.x/2, ACS_VLINE, pantalla.y);
attroff(COLOR_PAIR(2));

attron(COLOR_PAIR(3));
mvprintw(pelota.y, pelota.x, "o");
attroff(COLOR_PAIR(3));

attron(COLOR_PAIR(4));
for(i=-1; i<2; i++){
    mvprintw(barra_izquierda.y+i, barra_izquierda.x, "|");
    mvprintw(barra_derecha.y+i, barra_derecha.x, "|");
}
attroff(COLOR_PAIR(4));
}
}
```

## 2.2. Ejecución pong.c.

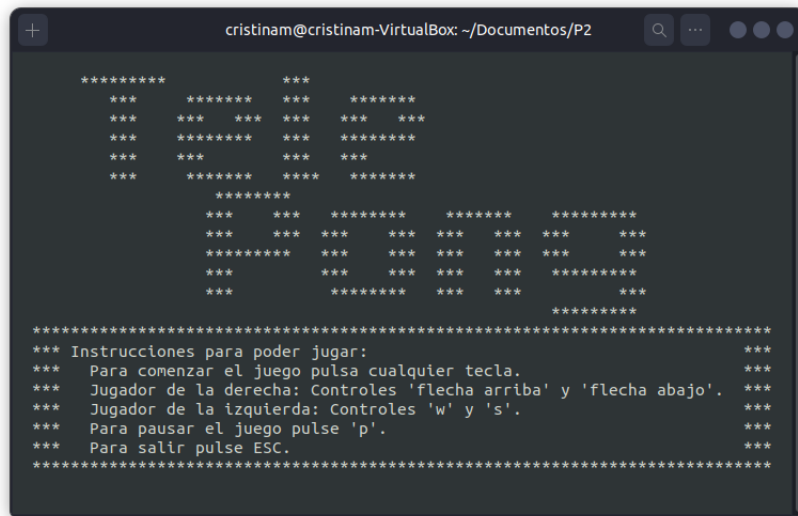


Figura 5: Página principal.

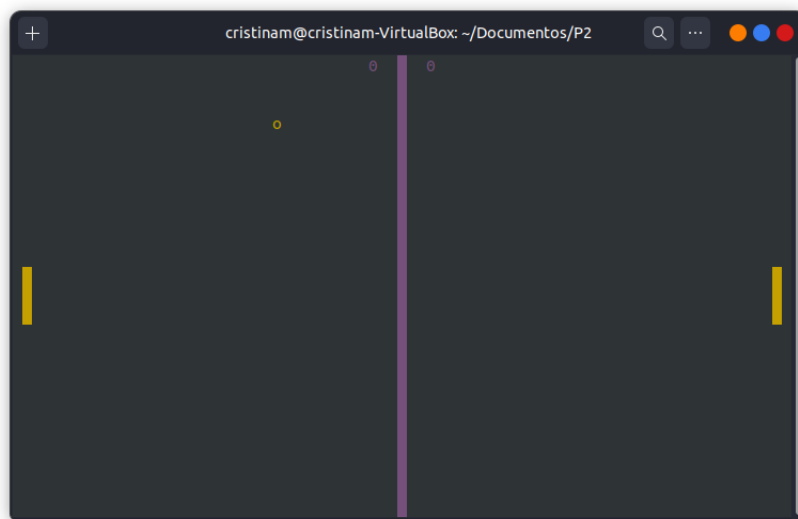


Figura 6: Principio del juego.

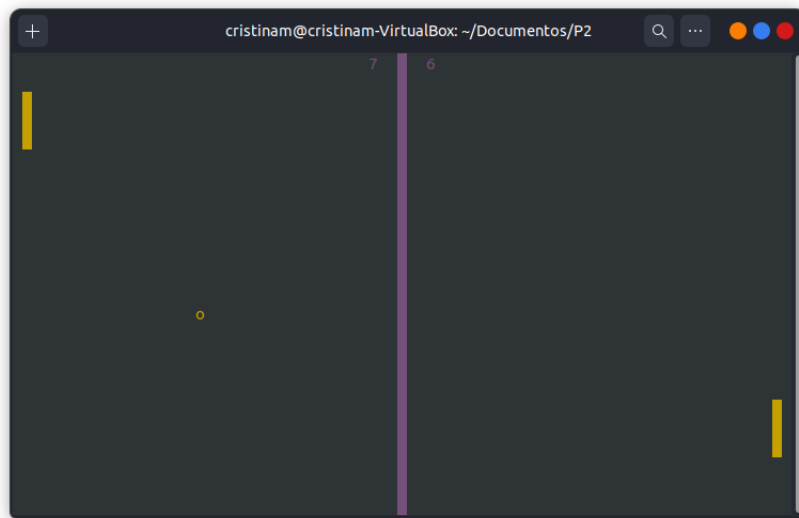


Figura 7: Juego.

Como hemos comentado en el apartado anterior, al inicio del programa se guarda el tamaño que tiene la terminal y adapta el juego al tamaño del mismo como podemos ver en la siguiente imagen (la imagen tiene la misma proporción que la imagen anterior, por lo tanto, se aprecia que el programa se ha adaptado al tamaño del terminal).

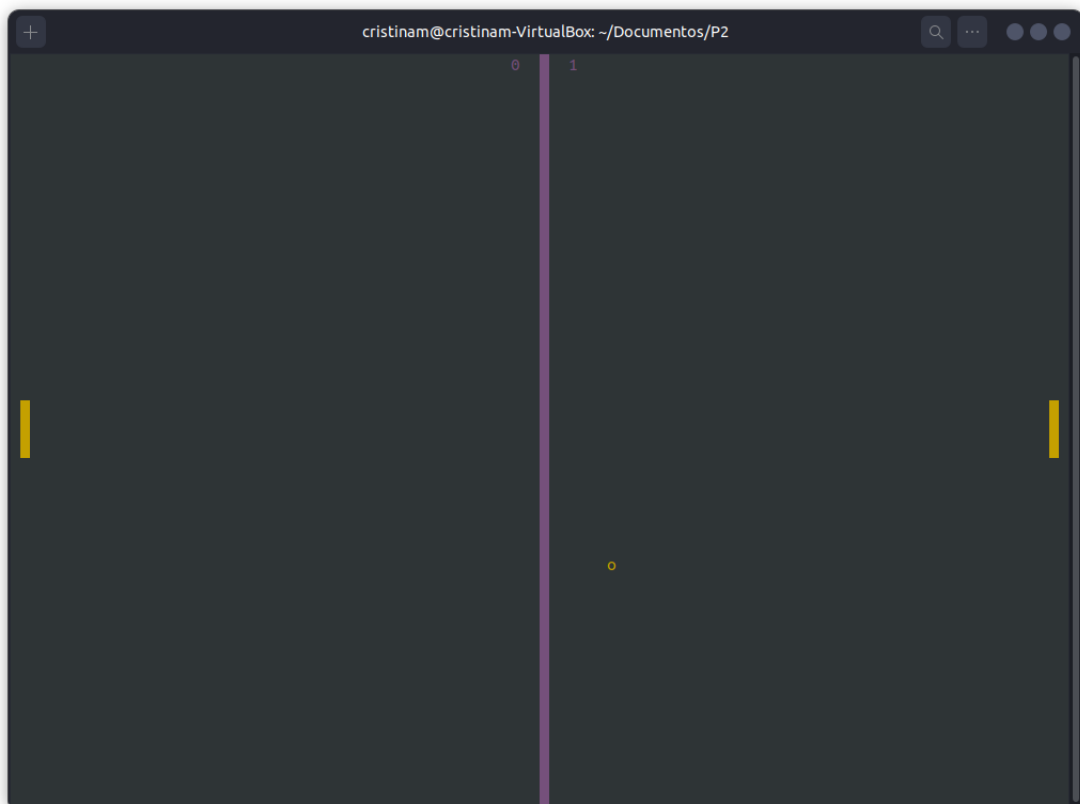


Figura 8: Aumentar pantalla.