



MEMORIA Y DOCUMENTACIÓN

PRÁCTICA 2

RAFAEL DELGADO GARCÍA-VALDECASAS

PABLO RIENDA SÁNCHEZ

GRADO EN INGENIERÍA INFORMÁTICA

Curso 2023-2024

1. Instalación ncurses y comprobación

Lo primero que hicimos fue instalar la librería ncurses.

Aquí vemos como ya están las librerías instaladas:

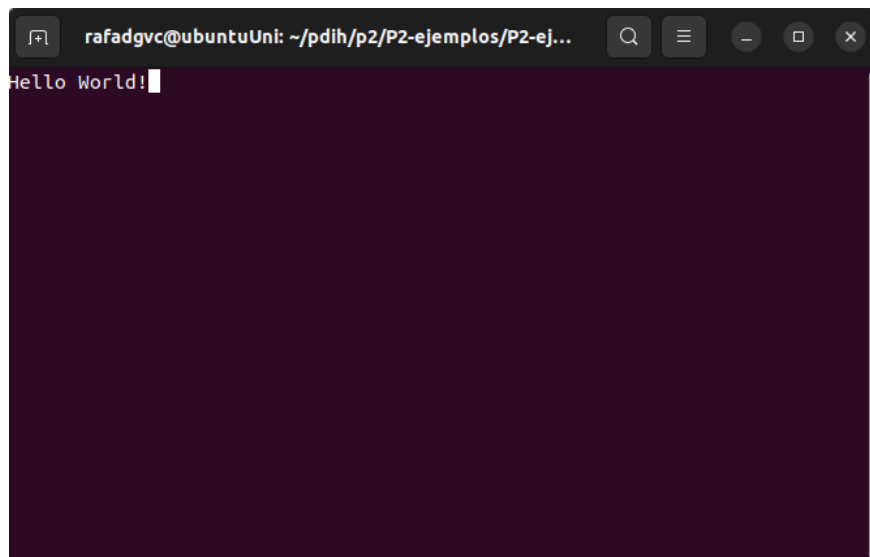
```
rafadgvc@ubuntuUni: ~/pdih/p2/P2-ejemplos/P2-ejemplos$ sudo apt-get install libncurses5-dev libncursesw5-dev
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
libncurses5-dev ya está en su versión más reciente (6.3-2ubuntu0.1).
libncursesw5-dev ya está en su versión más reciente (6.3-2ubuntu0.1).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 153 no actualizados.
rafadgvc@ubuntuUni: ~/pdih/p2/P2-ejemplos/P2-ejemplos$
```

Probamos el programa hello.c.

Lo compilamos y ejecutamos así:


```
rafadgvc@ubuntuUni: ~/pdih/p2/P2-ejemplos/P2-ejemplos$ gcc hello.c -o hello -lncurses
rafadgvc@ubuntuUni: ~/pdih/p2/P2-ejemplos/P2-ejemplos$ ./hello
rafadgvc@ubuntuUni: ~/pdih/p2/P2-ejemplos/P2-ejemplos$
```

Este es el resultado de la ejecución:

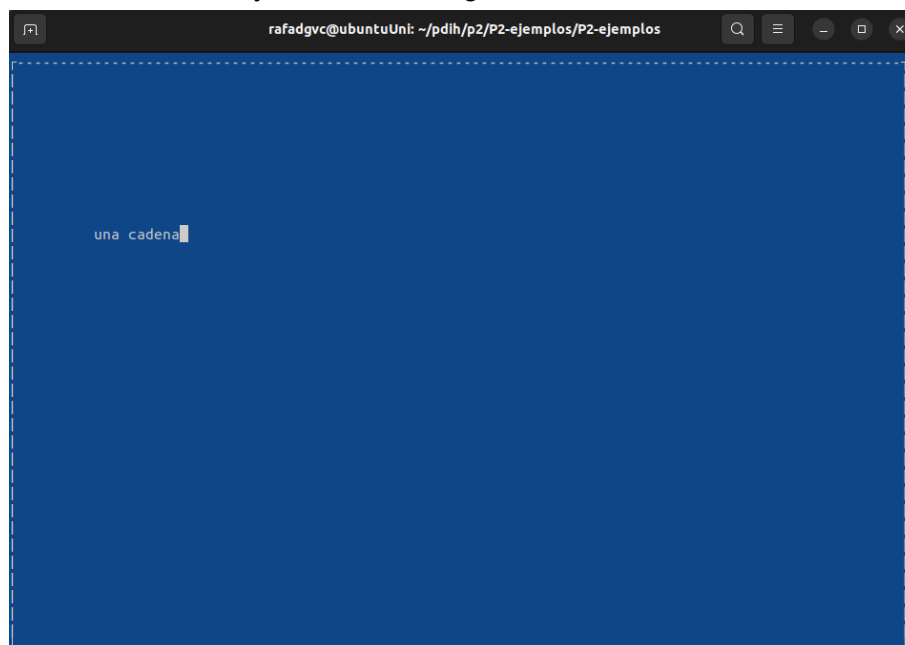
A terminal window with a dark purple background. The title bar shows the user 'rafadgvc@ubuntuUni' and the current directory '~/pdih/p2/P2-ejemplos/P2-ej...'. The terminal displays the text 'Hello World!' followed by a cursor.

Ahora probamos el ventana.c.

Lo compilamos y ejecutamos así:

A terminal window with a dark purple background. The title bar shows the user 'rafadgvc@ubuntuUni' and the current directory '~/pdih/p2/P2-ejemplos/P2-ejemplos'. The terminal shows three lines of commands and their outputs: 'gcc ventana.c -o ventana -lncurses', './ventana', and the prompt 'rafadgvc@ubuntuUni: ~/pdih/p2/P2-ejemplos/P2-ejemplos\$'.

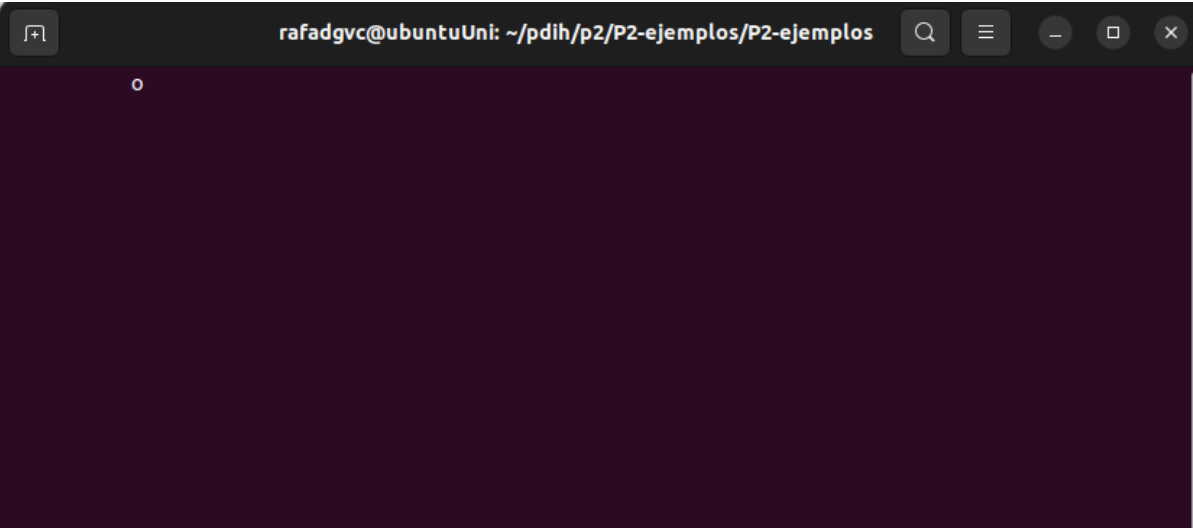
El resultado de la ejecución es el siguiente:

A terminal window with a dark purple background. The title bar shows the user 'rafadgvc@ubuntuUni' and the current directory '~/pdih/p2/P2-ejemplos/P2-ejemplos'. The terminal displays the text 'una cadena' followed by a cursor. The entire terminal content is enclosed in a dashed blue border.

Por último, compilamos y ejecutamos `pelotita.c`:



```
rafadgvc@ubuntuUni: ~/pdih/p2/P2-ejemplos/P2-ejemplos
rafadgvc@ubuntuUni:~/pdih/p2/P2-ejemplos/P2-ejemplos$ gcc pelletita.c -o pelletita -lncurses
rafadgvc@ubuntuUni:~/pdih/p2/P2-ejemplos/P2-ejemplos$ ./pelotita
rafadgvc@ubuntuUni:~/pdih/p2/P2-ejemplos/P2-ejemplos$
```

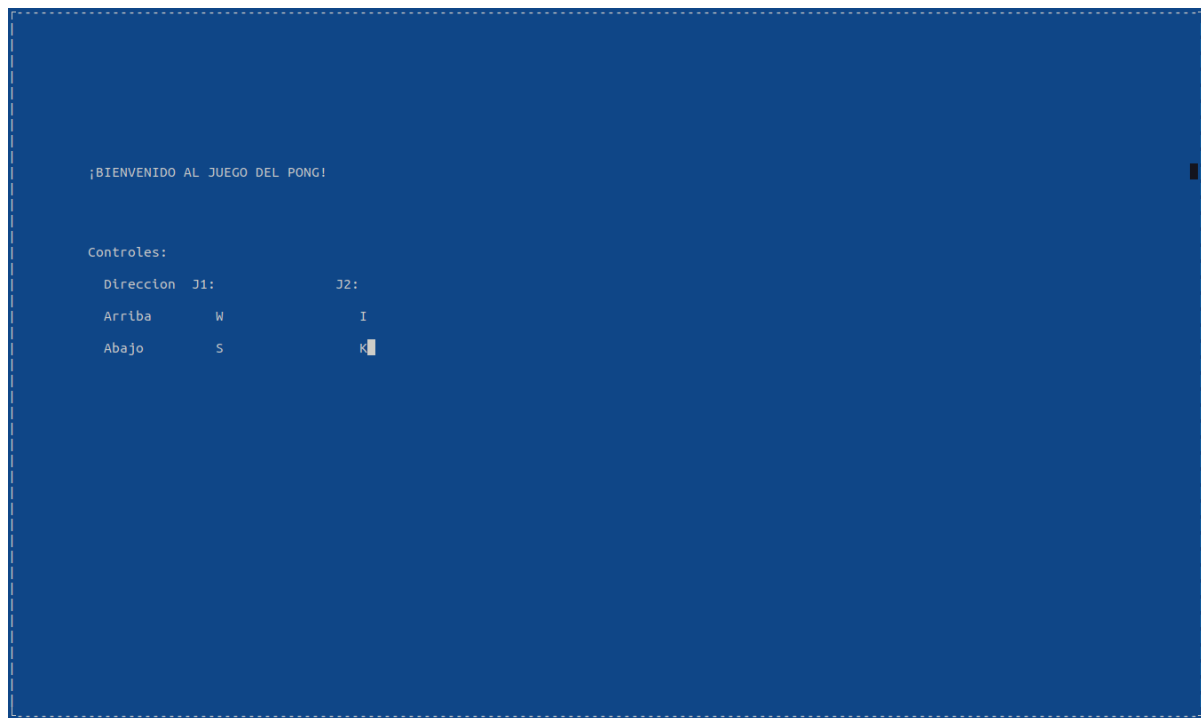


```
rafadgvc@ubuntuUni: ~/pdih/p2/P2-ejemplos/P2-ejemplos
o
```

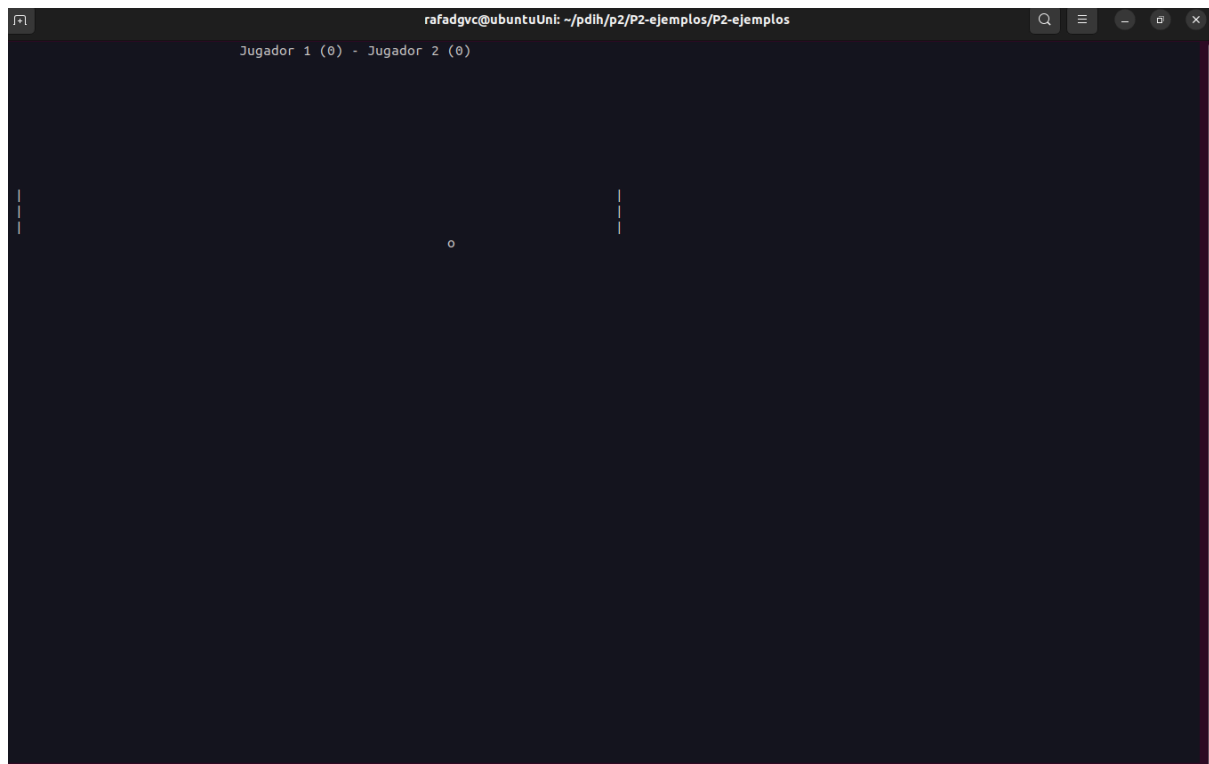
2. Creación del juego

En este apartado crearemos el juego tipo pong.

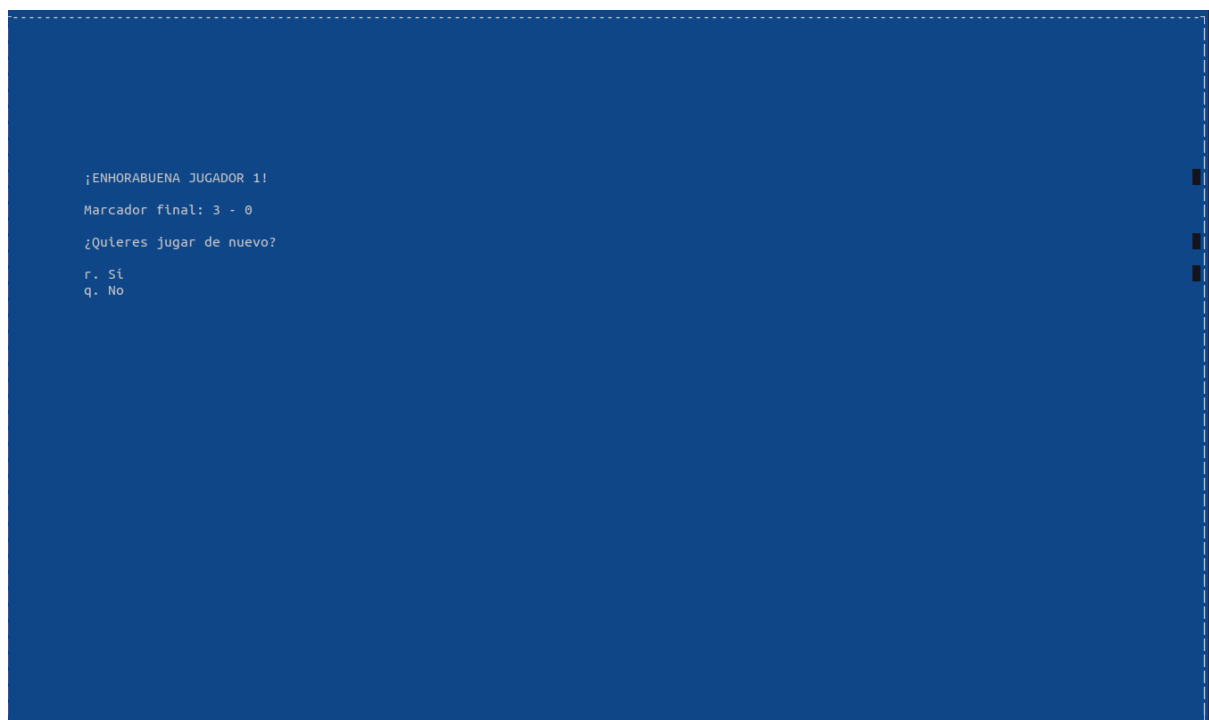
Para empezar, tal y como se nos indica en el ejercicio extra comenzamos con la pantalla de bienvenida:



El aspecto del juego sería el siguiente.



Y el menú de final de partida este:



El código final sería este:

Se recomienda que se compile y ejecute el programa de la siguiente manera para poder probar y jugar al juego.

```
rafadgvc@ubuntuUni: ~/pdih/p2/P2-ejemplos/P2-ejemplos$ gcc pong.c -o pong -lncurses
```

```
rafadgvc@ubuntuUni: ~/pdih/p2/P2-ejemplos/P2-ejemplos$ ./pong
```

```

#include <ncurses.h>

#define WIDTH 80
#define HEIGHT 20

#define PADDLE1_X 2
#define PADDLE2_X (WIDTH - 3)

// Variables globales
int ballX, ballY;
int ballDirectionX, ballDirectionY;
int paddle1Y, paddle2Y;
int scorePaddle1, scorePaddle2;

// Inicializa el juego
void init() {
    initscr();
    clear();
    noecho();
    cbreak();
    keypad(stdscr, TRUE);
    curs_set(0);
    nodelay(stdscr, TRUE); // Hace que getch() sea no bloqueante

    ballX = WIDTH / 2;
    ballY = HEIGHT / 2;
    ballDirectionX = 1;
    ballDirectionY = 1;
    paddle1Y = paddle2Y = HEIGHT / 2;
    scorePaddle1 = scorePaddle2 = 0;
}

void welcome(){
    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, "¡BIENVENIDO AL JUEGO DEL PONG!");
mvwprintw(window, 15, 10, "Controles:");
mvwprintw(window, 17, 12, "Direccion J1:          J2:");
mvwprintw(window, 19, 12, "Arriba    W          I");
mvwprintw(window, 21, 12, "Abajo      S          K");
wrefresh(window);

getch();
}

void gameover(){
    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, '|', '-');

    if (scorePaddle1 > scorePaddle2){
        mvwprintw(window, 10, 10, "¡ENHORABUENA JUGADOR 1!");
    }
    else{

```

```

mvwprintw(window, 10, 10, "¡ENHORABUENA JUGADOR 2!");
}

mvwprintw(window, 12, 10, "Marcador final: %d - %d", scorePaddle1, scorePaddle2);
mvwprintw(window, 14, 10, "¿Quieres jugar de nuevo?");
mvwprintw(window, 16, 10, "r. Sí");
mvwprintw(window, 17, 10, "q. No");
wrefresh(window);

int choice = getch();
while (choice != 'q' && choice != 'r') {
    choice = getch();
}

if (choice == 'r') {
    // Reinicia el juego
    init();
} else if (choice == 'q') {
    // Salir del juego
    endwin();
    exit(0);
}

}

void draw() {
    clear();

    // Dibujar la pelota
    mvprintw(ballY, ballX, "o");

    // Dibujar las palas
    for (int i = -1; i <= 1; i++) {
        mvprintw(paddle1Y + i, PADDLE1_X, "|");
        mvprintw(paddle2Y + i, PADDLE2_X, "|");
    }

    mvprintw(0, WIDTH / 2 - 10, "Jugador 1 (%d) - Jugador 2 (%d)", scorePaddle1,
scorePaddle2);

    refresh();
}

// Actualiza la posición de la pelota y detecta colisiones
void update() {
    // Actualiza la posición de la pelota

```

```

ballX += ballDirectionX;
ballY += ballDirectionY;

// Detecta colisiones con las palas
if (ballX == PADDLE1_X + 1 && ballY >= paddle1Y - 1 && ballY <= paddle1Y + 1) {
    ballDirectionX = 1;
}
if (ballX == PADDLE2_X - 1 && ballY >= paddle2Y - 1 && ballY <= paddle2Y + 1) {
    ballDirectionX = -1;
}

// Detecta colisiones con los bordes superior e inferior
if (ballY == 0 || ballY == HEIGHT - 1) {
    ballDirectionY *= -1;
}

// Detecta colisiones con los bordes laterales
if (ballX == 0 || ballX == WIDTH - 1) {
    // Aumenta el puntaje del jugador correspondiente
    if (ballX == 0) {
        scorePaddle2++;
    } else {
        scorePaddle1++;
    }
}

// Si uno de los jugadores llega a 3 goles, termina el juego
if (scorePaddle1 == 3 || scorePaddle2 == 3) {
    gameover();
}

// Reubica la pelota al centro del campo de juego
ballX = WIDTH / 2;
ballY = HEIGHT / 2;
}

// Maneja la entrada del jugador
void input() {
    int key = getch();

    // Controla la paleta del jugador 1 (izquierda)
    if (key == 'w' && paddle1Y > 1) {
        paddle1Y--;
    }
    if (key == 's' && paddle1Y < HEIGHT - 3) {
        paddle1Y++;
    }
}

```

```

        // Controla la paleta del jugador 2 (derecha)
        if (key == 'i' && paddle2Y > 1) {
            paddle2Y--;
        }
        if (key == 'k' && paddle2Y < HEIGHT - 3) {
            paddle2Y++;
        }
    }

int main() {
    // Pantalla de bienvenida
    welcome();
    // Inicializa el juego
    init();

    // Loop principal del juego
    while (1) {
        // Dibuja los objetos
        draw();

        // Maneja la entrada del jugador
        input();

        // Actualiza la posición de la pelota y detecta colisiones
        update();
        usleep(80000);
    }

    return 0;
}

```