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

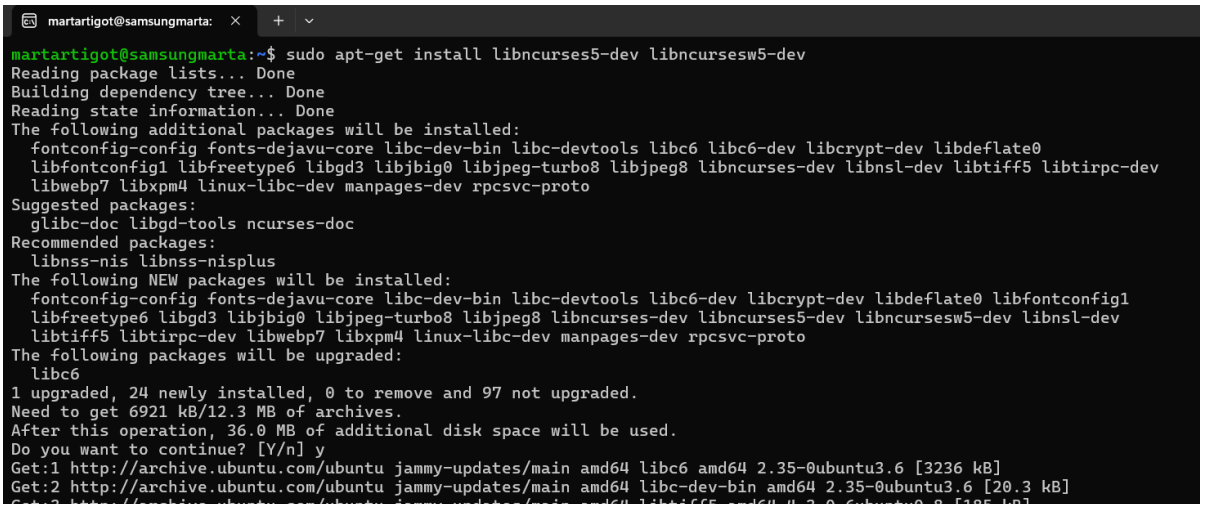
UNIVERSIDAD DE GRANADA

Marta Díaz Artigot

1. Requisitos Mínimos

1.1. Instalación de la librería ncurses

Para instalar la librería ncurses basta con utilizar el comando que aparece en el guión de prácticas.



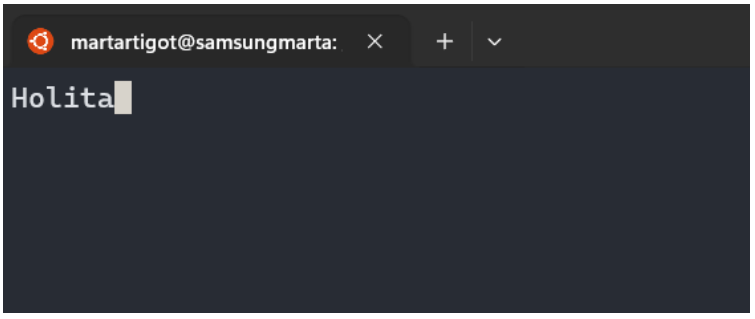
1.2. Programas de ejemplo

Como el código de los programas de ejemplo aparece en el guión de prácticas sólo incluiré capturas del funcionamiento.

Para compilar todos los programas utilizaré la orden: gcc programa.c -o programa -lncurses

1.2.1. Hello.c

```
martartigot@samsungmarta:/mnt/c/Users/marta/OneDrive/Documentos/PDIH$ gcc hello.c -o hello -lncurses
martartigot@samsungmarta:/mnt/c/Users/marta/OneDrive/Documentos/PDIH$ ./hello
```

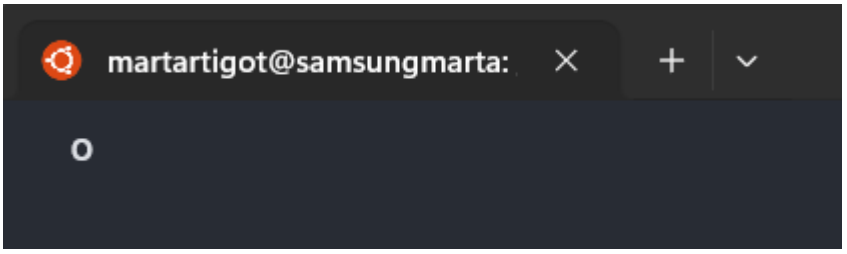


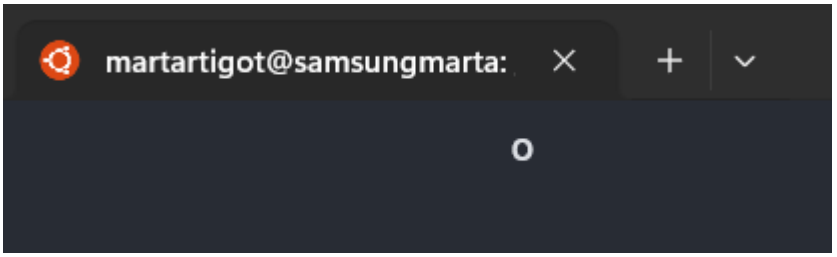
1.2.2. Ventana.c

```
martartigot@samsungmarta:/mnt/c/Users/marta/OneDrive/Documentos/PDIH/P2-ejemplos$ gcc ventana.c -o ventana -lncurses
martartigot@samsungmarta:/mnt/c/Users/marta/OneDrive/Documentos/PDIH/P2-ejemplos$ ./ventana
```

1.2.3. Pelotita.c

```
martartigot@samsungmarta:/mnt/c/Users/marta/OneDrive/Documentos/PDIH/P2-ejemplos$ gcc pelotita.c -o pelotita -lncurses
martartigot@samsungmarta:/mnt/c/Users/marta/OneDrive/Documentos/PDIH/P2-ejemplos$ ./pelotita
```





1.3. Juego tipo “Pong”

Desarrollo de un programa del mismo estilo que el videojuego PONG.

En esta primera captura se ven las definiciones que he utilizado y la declaración de las funciones.

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

// Definiciones
#define JUGADOR "|"
#define BOLA "o"
#define DELAY 90000

// Pares de colores
#define MARCADOR 2
#define CAMPO 1
#define JB 4


// Funciones
void mostrarPantallaInicial(int rows, int cols);
void mostrarPantallaDeJuego(int rows, int cols);
void actualizarPantallaDeJuego(WINDOW *window, int rows, int cols, int yJ1, int yJ2, int xBola, int yBola, int puntosJ1, int puntosJ2);
void mostrarPantallaDeVictoria(WINDOW *window,int rows, int cols, int puntosJ1, int puntosJ2);
int main();
```

1.3.1 mostrarPantallaInicial

La función crea una ventana con el título "PONG" en arte ASCII, controles y reglas del juego. Cuando el jugador pulsa una tecla, la ventana pasa a la ventana de juego.

```c
// Funciones desarrolladas
void mostrarPantallaInicial(int rows, int cols) {
    // Crear ventana para la pantalla inicial
    WINDOW *window = newwin(rows, cols, 0, 0);
    wbkgd(window, COLOR_PAIR(CAMPO));
    box(window, '|', '-');

    // Título "PONG" en arte ASCII
    mvprintw(2, cols / 2 - 25, ".--------.     ,------.    ,---.   .--.  .-_'''-.");
    mvprintw(3, cols / 2 - 25, "\\  _(`)_ \\  .'  .-,  '.  |    \\  |  | '_( )_   \\");
    mvprintw(4, cols / 2 - 25, "| (_ o._)| / ,-.|  \\ _ \\ |  ,  \\ |  ||(_ o _)|  '");
    mvprintw(5, cols / 2 - 25, "|  (_,_) /;  \\  '_ /  | :|  |\\_ \\|  |. (_,_)/___|");
    mvprintw(6, cols / 2 - 25, "|   '-.-' |  _`,/ \\ _/  ||  _( )_\\\\  ||  |  .------. ");
    mvprintw(7, cols / 2 - 25, "|   |     : (  '\\_/ \\\\   ;| (_ o _)  |'  \\\\  '-   .' ");
    mvprintw(8, cols / 2 - 25, "|   |      \\\\ `\"/  \\\\  ) / |  (_,_)\\\\  | \\\\  `-'`   | ");
    mvprintw(9, cols / 2 - 25, "/   )       '. \\\\_/``\".'  |  |    |  | \\\\       /");
    mvprintw(10, cols / 2 - 25, "`---'         '-----'    '--'    '--'   `'-...-' ");


    mvprintw(12, cols / 2 - 7, "CONTROLES");
    mvprintw(14, cols / 2 - 6, "Jugador 1");
    mvprintw(16, cols / 2 - 7, "Arriba -> w");
    mvprintw(17, cols / 2 - 7, "Abajo  -> s");
    mvprintw(19, cols / 2 - 6, "Jugador 2");
    mvprintw(21, cols / 2 - 7, "Arriba -> u");
    mvprintw(22, cols / 2 - 7, "Abajo  -> j");


    mvprintw(25, cols / 2 - 5, "REGLAS");
    mvprintw(27, cols / 2 - 18, "Gana el primero que llegue a 2 puntos.");

    // Mostrar nombre
    mvprintw(rows - 1, cols / 2 - 16, "Realizado por: Marta Díaz Artigot");
```

```
    // Actualizar la pantalla
    refresh();
    getch();

    // Eliminar la ventana
    delwin(window);
}
```

1.3.2. mostrarPantallaDeJuego

La función crea una ventana donde se desarrolla el juego de Pong. Define las posiciones iniciales de los jugadores y la bola, junto con las variables para los puntos de cada jugador. En un bucle, actualiza continuamente la pantalla con la posición de los elementos y verifica las colisiones. También captura las entradas del jugador para mover los paneles. Cuando uno de los jugadores alcanza los 3 puntos, muestra la pantalla de victoria y luego cierra la ventana del juego.

```
void mostrarPantallaDeJuego(int rows, int cols) {
    // Crear ventana para la pantalla de juego
    WINDOW *window = newwin(rows, cols, 0, 0);
    wbkgd(window, COLOR_PAIR(CAMPO));
    box(window, '|', '-');

    // Variables del juego
    int yJ1 = rows / 2 - 1, yJ2 = rows / 2 - 1;
    int xBola = cols / 2, yBola = rows / 2;
    int puntosJ1 = 0, puntosJ2 = 0;

    // Dirección de la bola
    int dirYBola = 1, dirXBola = -1;

    // Bucle del juego
    while (puntosJ1 < 3 && puntosJ2 < 3) {
        // Actualizar la pantalla de juego
        actualizarPantallaDeJuego(window, rows, cols, yJ1, yJ2, xBola, yBola, puntosJ1, puntosJ2);
```

```
    // Capturar la entrada del jugador
    int input = wgetch(window);
    nodelay(window, true);
    switch (input) {
        case 'w':
        case 'W':
            // Mover jugador 1 hacia arriba
            if (yJ1 > 1) {
                yJ1--;
            }
            break;
        case 's':
        case 'S':
            // Mover jugador 1 hacia abajo
            if (yJ1 < rows - 5) {
                yJ1++;
            }
            break;
        case 'i':
        case 'I':
            // Mover jugador 2 hacia arriba
            if (yJ2 > 1) {
                yJ2--;
            }
            break;
        case 'k':
        case 'K':
            // Mover jugador 2 hacia abajo
            if (yJ2 < rows - 5) {
                yJ2++;
            }
            break;
        default:
            break;
    }
```

```
    // Actualizar la posición de la bola
    yBola += dirYBola;
    xBola += dirXBola;

    // Verificar colisiones con los bordes superior e inferior
    if (yBola <= 1 || yBola >= rows - 2) {
        dirYBola = -dirYBola; // Invertir la dirección vertical
    }

    // Verificar colisiones con los jugadores
    if ((xBola == 3 && yBola >= yJ1 && yBola <= yJ1 + 1) ||
        (xBola == cols - 4 && yBola >= yJ2 && yBola <= yJ2 + 1)) {
        dirXBola = -dirXBola; // Invertir la dirección horizontal
    }

    // Verificar colisión con los bordes izquierdo y derecho
    if (xBola <= 2) {
        puntosJ2++; // Punto para el jugador 2
        yBola = rows / 2; // Reiniciar la posición de la bola
        xBola = cols / 2;
        dirXBola = -dirXBola; // Reiniciar la dirección de la bola
        usleep(1000000); // Esperar un segundo antes de continuar
    } else if (xBola >= cols - 3) {
        puntosJ1++; // Punto para el jugador 1
        yBola = rows / 2; // Reiniciar la posición de la bola
        xBola = cols / 2;
        dirXBola = -dirXBola; // Reiniciar la dirección de la bola
        usleep(1000000); // Esperar un segundo antes de continuar
    }

    // Usar usleep para la velocidad del juego
    usleep(DELAY);
}
```

```
    // Mostrar pantalla de victoria
    mostrarPantallaDeVictoria(window, rows, cols, puntosJ1, puntosJ2);

    // Eliminar la ventana
    delwin(window);
}
```

1.3.3. actualizarPantallaDeJuego

La función se encarga de actualizar la ventana de juego con la posición de los elementos en cada iteración del bucle principal. Borra la pantalla y dibuja el campo, la bola y los jugadores. También muestra el marcador de puntos para ambos jugadores en la parte superior de la ventana. Finalmente, actualiza la pantalla para reflejar los cambios realizados.

```c
void actualizarPantallaDeJuego(WINDOW *window, int rows, int cols, int yJ1, int yJ2, int xBola, int yBola, int puntosJ1, int puntosJ2) {
    // Borrar la pantalla
    werase(window);

    // Dibujar el campo y los elementos del juego
    wbkgd(window, COLOR_PAIR(CAMPO));
    box(window, '|', '-');
    wattron(window, COLOR_PAIR(JB));
    mvwprintw(window, yBola, xBola, BOLA);
    mvwprintw(window, yJ1, 2, JUGADOR);
    mvwprintw(window, yJ2, cols - 3, JUGADOR);
    wattroff(window, COLOR_PAIR(JB));

    // Mostrar marcador
    wattron(window, COLOR_PAIR(MARCADOR));
    mvwprintw(window, 2, 25, "J1: %d", puntosJ1);
    mvwprintw(window, 2, cols - 30, "J2: %d", puntosJ2);
    wattroff(window, COLOR_PAIR(MARCADOR));

    // Actualizar la pantalla
    wrefresh(window);
}
```

1.3.4. mostrarPantallaDeVictoria

Esta función muestra la pantalla final del juego. Borra la ventana y la redibuja con el mensaje de victoria y el marcador final en el centro. También indica cómo volver a jugar o salir. Si el usuario elige jugar de nuevo, reinicia el juego.

```c
void mostrarPantallaDeVictoria(WINDOW *window, int rows, int cols, int puntosJ1, int puntosJ2) {
    // Crear ventana para la pantalla de victoria
    // WINDOW *window = newwin(rows, cols, 0, 0);
    werase(window);
    wbkgd(window, COLOR_PAIR(CAMPO));
    box(window, '|', '-');

    // Mostrar mensaje de victoria
    if (puntosJ1 > puntosJ2) {
        mvwprintw(window, rows / 2 - 3, cols / 2 - 12, "¡El jugador J1 ha ganado!");
    } else {
        mvwprintw(window, rows / 2 - 3, cols / 2 - 12, "¡El jugador J2 ha ganado!");
    }

    // Mostrar marcador final
    mvwprintw(window, rows / 2, cols / 2 - 4, "MARCADOR");
    mvwprintw(window, rows / 2 + 2, cols / 2 - 2, "%d - %d", puntosJ1, puntosJ2);



    // Mostrar mensaje para volver a jugar o terminar
    mvwprintw(window, rows - 2, cols / 2 - 20, "Pulsa 'v' para volver a jugar o 'q' para salir");

    // Actualizar la pantalla
    wrefresh(window);
```

```
    // Esperar hasta que el usuario presione 'v' o 'q'
    int input;
    while ((input = getch()) != 'v' && input != 'q') {}
    werase(window);
    // Si el usuario quiere volver a jugar, llamar a la función principal nuevamente
    if (input == 'v') {
        wrefresh(window);
        werase(window);
        main();
    }

    // Eliminar la ventana
    delwin(window);
}
```

1.3.5 main

```
int main() {
    // Inicialización de ncurses
    initscr();
    noecho();
    curs_set(FALSE);

    // Inicialización de colores
    start_color();
    init_pair(CAMPO, COLOR_WHITE, COLOR_BLACK);
    init_pair(MARCADOR, COLOR_BLUE, COLOR_BLACK);
    init_pair(JB, COLOR_CYAN, COLOR_BLACK);

    // Obtener tamaño de la ventana
    int rows, cols;
    getmaxyx(stdscr, rows, cols);

    // Mostrar pantalla inicial
    mostrarPantallaInicial(rows, cols);

    // Mostrar pantalla de juego
    mostrarPantallaDeJuego(rows, cols);

    // Fin de ncurses
    endwin();
    return 0;
}
```

2. Funcionamiento y Requisitos Ampliados

2.1. Requisitos Ampliados

Los requisitos ampliados están incluidos en las capturas de pantalla de código anteriores. Son la función mostrarPantallaDeVictoria y mostrarPantallaInicial.

2.2 Funcionamiento

Pantalla inicial:

Juego:

Pantalla final: