

## Práctica 2



**UNIVERSIDAD  
DE GRANADA**

## Instalación de Ncurses

Para mi práctica voy a utilizar el subsistema de Linux en Windows (WSL) y lo primero es instalar las librerías necesarias.

Al ser Ubuntu utilizo el comando *sudo apt-get install libncurses5-dev libncursesw5-dev* y se instala correctamente.

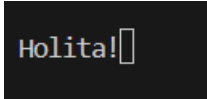
### Ejemplo Sencillo

El primer ejemplo es simplemente sacar una cadena de texto por pantalla utilizando ncurses para comprobar que funciona y compila correctamente.

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

*Initscr()*; Inicializa el modo curses.  
*Printw()*; Imprime por pantalla la cadena de texto indicada.  
*Refresh()*; Muestra el contenido por pantalla.  
*Getch()*; Captura un carácter por teclado.  
*Endwin()*; Termina el modo curses.

Compilamos con el comando *gcc hello.c -o hello -lncurses* y ejecutamos. El resultado es el siguiente:



### Ejemplo Ventana

El objetivo de este ejemplo es crear una ventana con bordes y distintos pares de colores. Primero se comprueba que el terminal tiene soporte de color y en caso contrario se muestra el mensaje por pantalla y se termina el programa.

Si tiene soporte para color entonces se inicia y se crean 3 pares distintos de colores, se indica primero el número identificador del par, el color de la letra y el color del fondo.

Con *getmaxyx()* obtenemos el tamaño máximo del terminal en el momento de la ejecución que se usará para crear la ventana.

```
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();

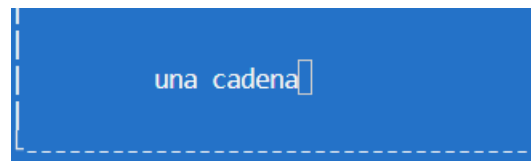
    refresh();
    getmaxyx(stdscr, rows, cols);
```

```
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;
```

Se crea el objeto de la ventana y con *wbkgd()* se establece el par de colores que serán utilizados. Además, con *box()*, se puede enmarcar la ventana con un borde. Una vez hecho se utiliza *mvprintw()* para imprimir por pantalla la cadena en la ventana escogida y *wrefresh()* refresca el contenido de la misma. Una vez compilado y ejecutado el resultado es el siguiente.



## Ejemplo Pelota

El objetivo de este ejemplo es colocar una pelota y hacerla moverse y rebotar por la ventana. Para conseguir esto se establecen unos máximos y mínimos entre los que la pelota puede moverse y se cambia de dirección el movimiento de esta.

```
#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();
}
```

Con *curs\_set()* se indica el tipo de cursor de forma que puede hacerse invisible. Después comienza un bucle infinito en el que se irá incrementando las coordenadas en las que se pinta la pelota.

Como condición para que rebote de lado a lado, se indica que si la siguiente coordenada sobrepasa el máximo o el mínimo indicado entonces se cambia la dirección de avance. El resultado se mostrará en clase para ver el movimiento.

## Pong.c

Como último programa he creado una versión simple del juego Pong. El primer paso es inicializar las variables que vamos a utilizar como el máximo de tamaño del tablero y la posición inicial de la pelota y jugadores.

```
#define DELAY 100000

int main(int argc, char *argv[]) {
    int rows, cols;
    int x = 10, y = 5;
    int max_y = 15, max_x = 70;
    int next_x = 0;
    int directionx = 1;
    int next_y = 0;
    int directiony = 1;
    int p1x = 5, p1y = 10, p2x = 65, p2y = 10;
    int ch = 0;
    bool play = true;

    initscr();
    start_color();
    init_pair(1, COLOR_BLACK, COLOR_WHITE);
    noecho();
    cbreak();
    curs_set(0);
    //nodelay(stdscr, TRUE);
    getmaxyx(stdscr, rows, cols);
```

Después se inicializa el modo curses, se define el par de colores y se establece el cursor invisible, *noecho()* para que no se pinten los caracteres que se van pulsando y *cbreak()* para que los caracteres no se procesen y estén disponibles inmediatamente para el programa.

Posteriormente se crea una nueva ventana de inicio en la que se mostrará el autor y las instrucciones. También se establece un marco para la ventana.

Aclaración: A pesar de que la ventana de inicio está hecha de la misma forma que la ventana de final del juego, la primera no funciona y tan solo se ve una pantalla en negro hasta que se pulsa la tecla y comienza el juego.

```
WINDOW * inicio = newwin(rows, cols, 0, 0);
wbkgd(inicio, COLOR_PAIR(1));
mvwprintw(inicio, 1, 1, "AUTOR: Pablo García Fernández");
mvwprintw(inicio, 2, 1, "Jugador 1: ARRIBA -> W, ABAJO -> S");
mvwprintw(inicio, 3, 1, "Jugador 2: ARRIBA -> Flecha Arriba, ABAJO -> Flecha Abajo");
mvwprintw(inicio, 4, 1, "Pulse cualquier tecla para empezar a jugar");
box(inicio, '|', '-');
wrefresh(inicio);
getch();
timeout(0);
```



Ahora se crea el bucle en el que se desarrolla el juego.

```
while(play) {
    clear();

    // Se dibujan los bordes del campo
    for (int i = 0; i < max_x; i++){
        mvprintw(0, i, "-"); //Pinta la linea inferior
        mvprintw(max_y, i, "-"); //Pinta la linea superior
    }
    for (int i = 1; i < max_y; i++){
        mvprintw(i, 0, "|"); //Pinta la linea izquierda
        mvprintw(i, max_x, "|"); //Pinta la linea derecha
    }
    //Pintamos la pelota y ambas raquetas
    mvprintw(y, x, "o");
    for (int i = 0; i < 4; i++){
        mvprintw(p1y + i, p1x, "|");
        mvprintw(p2y + i, p2x, "|");
    }
}
```

La variable booleana *play* se mantiene *true* hasta que al final el jugador decide parar el juego.

En dos bucles se pintan los bordes del campo con guiones y barras verticales. Luego se pinta la pelota y en otro bucle ambas raquetas de los jugadores.

```
ch = getch();
switch (ch)
{
case 'w':
    if(p1y > 1){
        p1y -= 1;
    }
    break;
case 's':
    if(p1y+4 < max_y){
        p1y += 1;
    }
    break;
case 65: //Flecha arriba
    if(p2y > 1){
        p2y -= 1;
    }
    break;
case 66: //flecha abajo
    if(p2y+4 < max_y){
        p2y += 1;
    }
    break;
}
```

Una vez se ha pintado todo lo necesario se comienza a gestionar mediante un *switch* las pulsaciones de los jugadores en el teclado. Como se indica en las instrucciones el jugador 1 utiliza W y S y el jugador 2 la flecha arriba y abajo. Para controlar que las raquetas no sobrepasan los límites del campo se comprueban su posición.

```
usleep(DELAY);

next_x = x + directionx;
next_y = y + directiony;
```

Antes de ejecutar el movimiento de la pelota en la dirección indicada, se pone un retardo para que el juego vaya a una velocidad adecuada al procesamiento del resto del programa.

En el siguiente fragmento se muestra como se gestionan los rebotes tanto en los límites inferior y superior como en las raquetas de ambos jugadores. La forma de hacerlo es simplemente comprobar que la pelota coincide con las coordenadas de las raquetas o que va a superar los límites en el eje Y. En caso afirmativo se invierte la dirección de la pelota.

```
//La pelota rebota si toca en los límites superior o inferior
if (next_y >= max_y || next_y <= 0) {
    directiony*= -1;
} else {
    y+= directiony;
}

//La pelota rebota si da en alguna las raquetas de los jugadores
if ((next_y >= p1y && next_y <p1y+4) && next_x==p1x + 1) {
    directionx*= -1;
}

if ((next_y >= p2y && next_y <p2y+4) && next_x==p2x -1) {
    directionx*= -1;
}
```

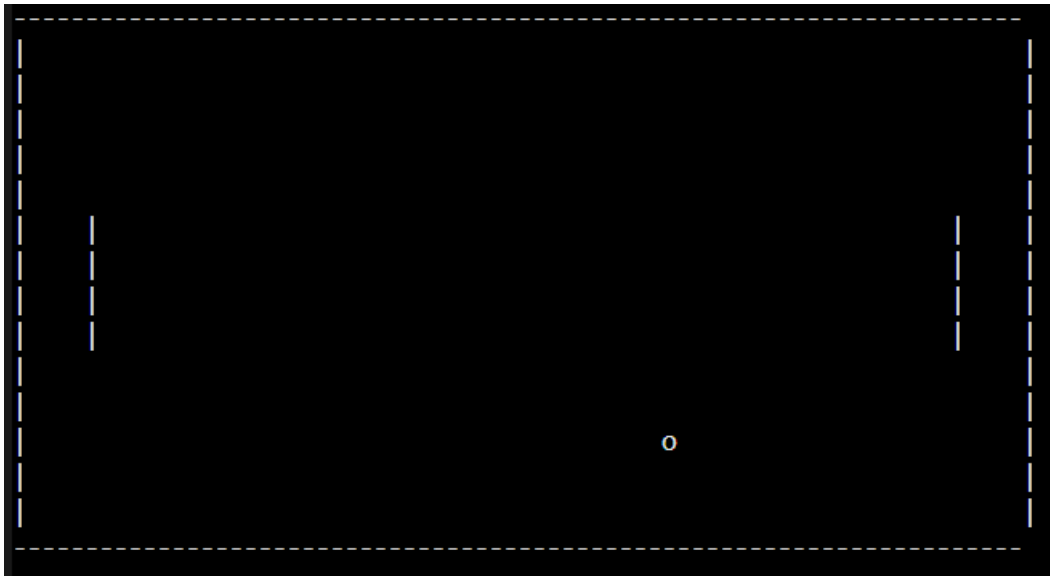
Por último, ya solo queda el código correspondiente a lo que ocurre cuando se marca un gol (se superan los límites en el eje X). Se crea una nueva ventana para el final del juego y en caso de que la tecla pulsada sea la F entonces se finaliza el bucle y termina el programa, en caso contrario se continúa jugando hasta el siguiente gol.

```
//Si la pelota sale por el lado izquierdo o derecho , se pregunta si quiere continuar
if (next_x >= max_x || next_x < 0) {
    timeout(-1);

    WINDOW * final = newwin(rows, cols, 0, 0);
    wbkgd(final, COLOR_PAIR(1));
    mvwprintw(final, 5, 10, "Se ha marcado un gol");
    mvwprintw(final, 6, 10, "Si quiere salir pulse F o pulse cualquier otra tecla para seguir jugando");
    box(final, '|', '-');
    wrefresh(final);

    char tecla = getch();
    if(tecla == 'f')
        play = false;
    else{
        timeout(0);
        x = 10;
        y = 5;
    }
} else {
    x+= directionx;
}
```

Una vez ejecutado se muestra de la siguiente forma:



Se ha marcado un gol  
Si quiere salir pulse F o pulse cualquier otra tecla para seguir jugando