UNIVERSIDAD
DE GRANADA

Periféricos y Dispositivos de Interfaz Humana

Práctica 2 – Uso de bibliotecas de programación de interfaces de usuario en modo texto - *ncurses*

Arturo Alonso Carbonero

ÍNDICE

1. Introducción

El objetivo de esta práctica es desarrollar el juego “Pong” en modo texto utilizando las funciones que ofrece la biblioteca ‘ncurses’ en lenguaje C. Para ello, he utilizado una máquina virtual con Ubuntu Server 20.04.1 (sin interfaz gráfica) en la que he configurado *SSH* para enviar el código de la práctica y ejecutarlo (para utilizar un editor más cómodo a la hora de programar que *vi* o *nano*).

En el repositorio de la práctica se encuentra esta memoria, el código de la practica (practica2.c), un directorio con capturas de algunos resultados y un vídeo que muestra el resultado completo de la práctica. Cabe aclarar que desconozco si la práctica funciona en todas las máquinas ya que el tamaño de la máquina virtual es fijo, aunque no debería haber problemas ya que los marcos de las ventanas los he fijado sin tener en cuenta esto (es decir, para cualquier máquina utilizando la función *getmaxyx()*). En cualquier caso, en el vídeo se muestra el resultado perfectamente.

2. Instalación y configuración de *ncurses*

Para instalar *ncurses* en la máquina mencionada anteriormente, he ejecutado el comando que se muestra en la imagen (versión para Ubuntu). No es necesario realizar nada más.

```
arturoac@arturoac:~$ sudo apt-get install libncurses5-dev libncursesw5-dev
```

Para compilar y ejecutar los programas utilizaré los comandos ***gcc archivo.c -lncurses -o archivo*** y ***./archivo*** respectivamente.

3. Ejemplos

A continuación se muestra el resultado de compilar y ejecutar los ejemplos del guion con el objetivo de comprobar si la biblioteca *ncurses* ha sido instalada con éxito.

3.1. hello.c

```
arturoac@arturoac:~/PDIH/P2$ gcc hello.c -lncurses -o hello
arturoac@arturoac:~/PDIH/P2$ ./hello
```

Compilación y ejecución

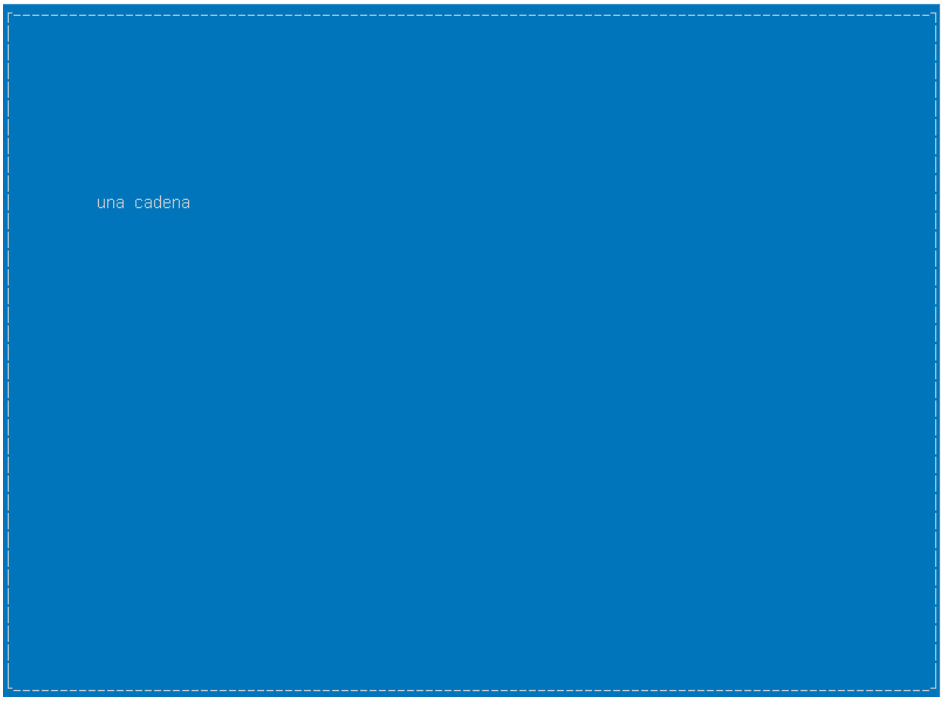
```
Hello World!
```

Resultado

3.2. ventana.c

```
arturoac@arturoac:~/PDIH/P2$ gcc ventana.c -lncurses -o ventana
arturoac@arturoac:~/PDIH/P2$ ./ventana
```

Compilación y ejecución



Resultado

3.3. pelotita.c

En este caso, el resultado no se aprecia en una imagen ya que la bola está en movimiento.

```
arturoac@arturoac:~/PDIH/P2$ gcc pelotita.c -lncurses -o pelotita
arturoac@arturoac:~/PDIH/P2$ ./pelotita
```

Compilación y ejecución

Resultado

4. practica2.c - Pong

4.1. Introducción

El juego cuenta con una pantalla de inicio con el título, las normas de juego y los controles, así como una pantalla para el proprio juego y dos más para mostrar el ganador y para abandonar el programa. Para ello he creado cuatro ventanas (*window* … *window4*) utilizando las funciones de *ncurses* tal y como se muestra en la siguiente imagen.

```
57    // Pantalla inicial
58    WINDOW *window=newwin(rows, cols, 0, 0);
59    wbkgd(window, COLOR_PAIR(CAMPO));
```

Ventana de inicio

Visualmente, los jugadores serán el carácter '|' y la bola será la letra 'o'. Tanto la pantalla de inicio como la del juego tendrán marco (función *box* de *ncurses*) con los caracteres de la imagen siguiente.

```
10   #define JUGADOR "|"
11   #define BOLA "o"
```

Jugadores y Bola

```
60    box(window, '|', '-');
```

Marco

De ahora en adelante se mostrarán únicamente, en este memoria, fragmentos puntuales del código ya que su extensión es considerable y el mismo se encuentra en el repositorio de la entrega.

En el código de la práctica hay comentarios que aclaran el funcionamiento de determinadas funciones de la biblioteca que he utilizado y que no están en el guion como *nodelay()*.

4.2. Funcionamiento

El funcionamiento es el típico del juego. Dos jugadores que se desplazan en el eje Y situados en los extremos de la pantalla deben evitar que una bola toque la pared de su lado. Dicha bola, en este caso, solo rebotará con 45º por simplicidad. Para ello, he utilizado variables de tipo *bool* que, cuando se activan (true) o desactivan (false), modifican las coordenadas siguientes de la bola.

```
178      // Sentido
179      if(diagonalDchaArriba){
180        x=sigX; y=sigY;
181        sigX++; sigY--;
182      }else if(diagonalDchaAbajo){
183        x=sigX; y=sigY;
184        sigX++; sigY++;
185      }else if(diagonalIzdaArriba){
186        x=sigX; y=sigY;
187        sigX--; sigY--;
188      }else{
189        x=sigX; y=sigY;
190        sigX--; sigY++;
191      }
```

Para que el juego funcione, todo el código del mismo sucede en el seno de un bucle que termina cuando uno de los jugadores alcanza 5 puntos. Cuando un jugador anota, aumenta en uno su puntuación y la bola comienza en el centro de la pantalla moviéndose desde el mismo hacia el lado del jugador que anotó. Al alcanzar 5 puntos, ocurre una pausa y el bucle finaliza.

```
311       // Fin del juego
312       if(puntosJ1==5 || puntosJ2==5){
313         getch();
314       }
```

Al desplazarse, la bola deja un rastro de ‘o’, por lo que he implementado una solución que, aunque bastante simple y “tosca”, soluciona tanto ese problema de aspecto como el de los rebotes, ya que también en estos casos sucedía algo similar. Básicamente, consiste en mostrar un espacio (“ ”) en las posiciones que no deben tener una ‘o’ en la iteración actual del bucle.

```
239       if(sigX<x && diagonalIzdaArriba){
240         if(x-1==0){
241           mvwprintw(window2, y+1, x+1, " ");
242         }else{
243           mvwprintw(window2, y+1, x+1, " ");
244           mvwprintw(window2, y, x+2, " ");
245           mvwprintw(window2, y-1, x+3, " ");
246
247           mvwprintw(window2, yJ2, xJ2-1, " ");
248         }
```

Ejemplo para una trayectoria

La trayectoria de la bola se modifica cuando esta choca con una pared o con un jugador, obteniendo la trayectoria directamente opuesta con una inclinación de 45º. Por ejemplo, pasa de subir hacia la derecha a bajar hacia la derecha (techo superior) o, en el caso de rebotar con un jugador (J2), de subir a la derecha a subir a la izquierda.

```
210       }else if(diagonalIzdaAbajo && y==36){ // Suelo
211         diagonalIzdaAbajo=false;
212         diagonalIzdaArriba=true;
213       }else if(diagonalIzdaArriba && (y==yJ1 && x==xJ1+1)){ // Jugador 1
214         diagonalIzdaArriba=false;
215         diagonalDchaArriba=true;
216         mvwprintw(window2, y+1, x+1, " "); // Aspecto al rebotar
```

Ejemplo de rebote con suelo y jugador

Además, durante la partida se muestra el marcador con los puntos de cada jugador. Para que aumente la puntuación, compruebo si la bola alcanza una posición en los extremos del eje X. Cuando esto sucede, la posición de la bola vuelve al centro de la pantalla y se realiza una pausa.

```
283       // Conteo de puntos
284       if(x==cols-1){
285         puntosJ1++;
286         y=yCentral; x=xCentral;
287         sigY=y; sigX=x;
288         getch();
289       }else if(x==0){
290         puntosJ2++;
291         y=yCentral; x=xCentral;
292         sigY=y; sigX=x;
293         getch();
294       }
295
296       // Mostrar marcador
297       wattron(window2, COLOR_PAIR(MARCADOR));
298       mvwprintw(window2, 2, 25, "J1: ");
299       char ptosJ1[50]; // Convertir int en const char*
300       sprintf(ptosJ1, "%d", puntosJ1); // --^
301       mvwprintw(window2, 2, 29, ptosJ1);
```

Conteo de puntos y marcador

Para el movimiento de los jugadores, se lee un carácter por teclado y se comprueba cuál se ha pulsado. El jugador (J1 o J2) se moverá en el sentido pertinente (en el eje Y) según los controles que se muestran en la pantalla de inicio. He utilizado la función *nodelay()* de la biblioteca para que la función *wgetch()* no realice una pausa. Sin ella, cada iteración del bucle sucedía con cada pulsación de tecla, por lo que no era posible jugar. De esta forma, el bucle procede (por lo que la bola se mueve) de forma independiente al movimiento de los jugadores.

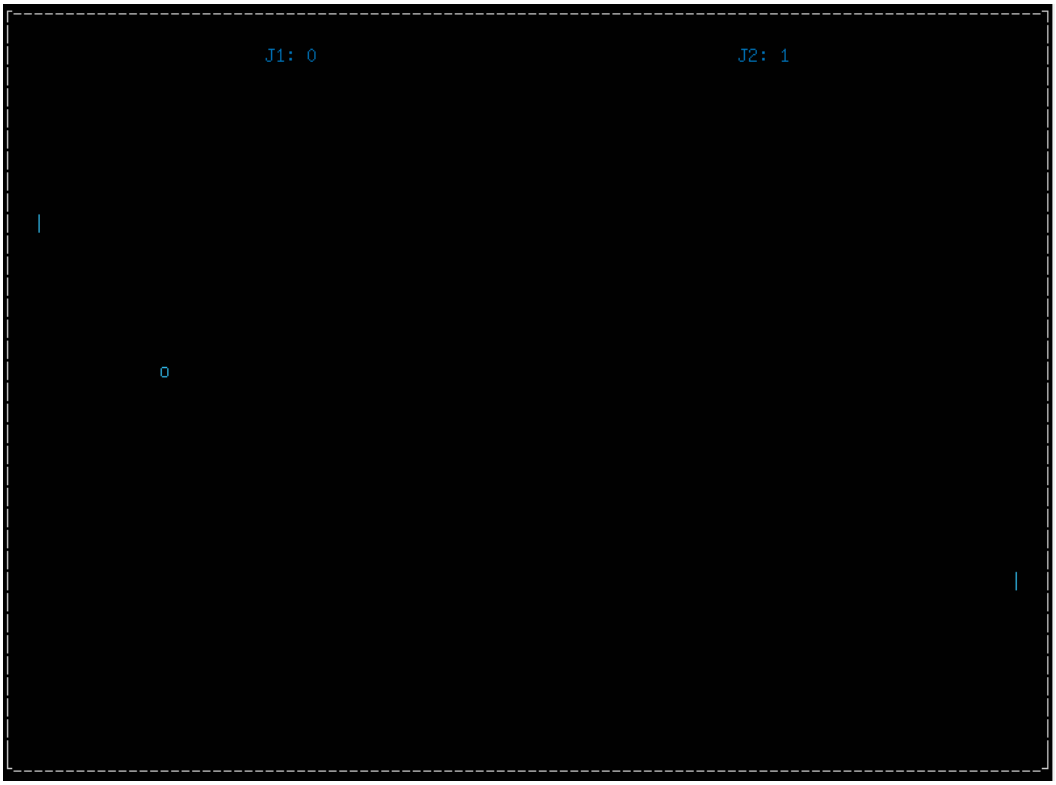
```
126       // Moviemiento de jugadores
127       ch=wgetch(window2); // Leer carácter de teclado
128       nodelay(window2, true); // getch() sin pausa
129       switch(ch){
130         case 'w':
131           if(yJ1>1){
132             wattron(window2, COLOR_PAIR(JB));
133             mvwprintw(window2, yJ1--, xJ1, JUGADOR); // Moviemiento
134             mvwprintw(window2, yJ1+1, xJ1, " "); // Aspecto
135             mvwprintw(window2, yJ1+1, xJ1+1, " "); // Aspecto
136             wattroff(window2, COLOR_PAIR(JB));
137             wrefresh(window2);
138           }
139           break;
```

nodelay() y ejemplo de movimiento

Cuando el juego finaliza se muestran dos pantallas tal y como se ha mencionado. Una para mostrar el jugador que ha obteniendo la victoria y otra para realizar una pausa extra para abandonar el juego.

4.3. Resultado

Pantalla Inicial



Pantalla del Juego