Examen de Informática Industrial
4º GITI (13/12/18)

PROBLEMA. TIEMPO: 1 HORA Y 45 MINUTOS (VALORACIÓN: 50%)

Resumen: Se pide realizar en C y con llamadas POSIX un programa multihilo que implementa un juego de ordenador. Se trata de acertar a un blanco móvil disparando desde un cañón que puede desplazarse hacia la izquierda o la derecha. El programa recibe comandos de movimiento y de disparo por un "socket" (**sí**) orientado a conexión. Las constantes simbólicas y declaraciones necesarias están definidas en la cabecera "**juego.h"**, que se supone disponible.

Especificación detallada:

- **Argumentos de la línea de comandos**: Los argumentos **de la línea de comandos** se interpretan como **cfin** (argumento 1, entero codificado en caracteres), **timout** (argumento 2, entero codificado en caracteres), **ciclo** (argumento 3, entero menor que 1000, codificado en caracteres) y **psocket** (argumento 4, entero codificado en caracteres).
- **"Socket" de comandos**: Es un "socket" **orientado a conexión** que se conecta en modo **pasivo** y utiliza para ello el puerto local **TCP psocket**. Se utiliza para recibir los comandos del juego en forma de estructuras de tipo **comando**, que indica si están o no pulsados cada uno de los tres pulsadores: "Izquierda", "Derecha" y "Fuego".
- **Pantalla**: La pantalla tiene **NC** columnas y **NF** filas de píxeles ordenadas como indica la figura.
- **Datos compartidos**: Contendrán **al menos** las coordenadas del blanco y el cañón, la velocidad del cañón y un contador de impactos.
- **Hilo receptor**: Se encarga de recibir comandos por el "socket" de comandos y de ejecutarlos:
  - Si está pulsado el pulsador "Izquierda", decrementa la velocidad del cañón en **INCV** (píxeles por ciclo).
  - Si está pulsado el pulsador "Derecha", incrementa la velocidad del cañón en **INCV** (píxeles por ciclo).
  - Si está pulsado el pulsador "Fuego" arranca un hilo de control de bala, si hay menos de **MAXB** hilos activos de este tipo; en caso contrario ignora el pulsador.
- **Hilo de control del cañón**: Se encarga de crear y mover el cañón hacia la izquierda y la derecha de la pantalla, manteniéndolo siempre en la fila **FC**. Cada **ciclo** milisegundos el hilo modifica la posición del cañón de acuerdo con su velocidad actual (entero en píxeles por ciclo). Inicialmente el cañón está situado en el punto central de la fila **FC**, y su velocidad es nula. El hilo se crea a partir de la función de arranque **h_gun**, que se supone disponible, por lo que **no se pide su código**, sólo explicar brevemente cómo se sincroniza y comunica con el resto de hilos. Es necesario pasar a **hgun** el valor de **ciclo** convertido a puntero genérico.
- **Hilos de control de bala**: Cada hilo de control de bala crea en el momento de arrancar la imagen de la bala en la posición actual del cañón (función **crear_bala**), y a continuación hace que se mueva (función **mover_bala**) en dirección a la parte superior de la pantalla actualizando cada **ciclo** milisegundos sus coordenadas con velocidad **VBALA (**entero en píxeles por ciclo). El hilo destruye la bala (función **dest_bala**) y acaba si ésta se sitúa en algún momento en las coordenadas del blanco, o bien si se sale de la pantalla. En el primer caso además incrementa el contador de impactos. **No** es necesario utilizar un temporizador POSIX 1003.1c para crear el ciclo.
- **Hilo de control de blanco**: Controla el movimiento del blanco. Se crea a partir de la función de arranque **h_blanco**, que se supone disponible, por lo que **no se pide su código**, sólo explicar brevemente cómo se sincroniza y comunica con el resto de hilos.
- **Condiciones de fin:** El programa acaba cuando el número de impactos alcanza **cfin**, o transcurren más de **timout** segundos desde la recepción del último comando. Antes de acabar debe esperar a que terminen normalmente todos los hilos de control de bala existentes, sin ejecutar más comandos.

Otras condiciones:

- La condición de fin basada en tiempo (**timout** segundos) debe implementarse con un **temporizador POSIX 1003.1b**.
- Se supone que las funciones **h_blanco**, **h_gun**, **crear_bala**, **mover_bala** y **dest_bala** ya existen y están disponibles en una biblioteca.
- El programa deberá funcionar **independientemente** de los valores concretos de los argumentos y las constantes simbólicas.
- Es **necesario utilizar mutex y variables de condición** según lo indicado en clase para gestionar el acceso a datos compartidos y las necesidades de sincronización entre hilos.
- No es necesario considerar tratamiento de errores, salvo los especificados en el enunciado.
- Puede suponerse que todas las llamadas POSIX son "async-safe" y "pthread-safe".
- Es preciso **explicar resumidamente** los puntos fundamentales del programa mediante pseudocódigo, comentarios o esquemas.

Fichero de cabecera "juego.h":

```
#define   NC      <valor numérico>
#define   NF      <valor numérico>
#define   FC      <valor numérico>
#define   INCV    <valor numérico>
#define   VBALA   <valor numérico>
#define   MAXB    <valor numérico>

/* Comando del juego */
struct comando {
    int izquierda;   /* Puls. izquierda (0/1) */
    int derecha;     /* Puls. derecha (0/1) */
    int fuego;       /* Puls. de disparo (0/1) */
};

/* Funciones (se suponen disponibles) */
/* Hilo de control del blanco */
void *h_blanco(void *p);

/* Hilo de control del cañón */
/* Argumento: Periodo del ciclo en milisegundos
              (entero convertido a void *) */
void *h_gun(void *p);

/* Crear bala en fila fil y columna col */
/* Devuelve el identificador de la bala */
int crear_bala(int fil, int col);

/* Destruir bala con identificador ibala */
void dest_bala(int ibala);

/* Mover bala con identificador ibala
   a la fila fil y columna col */
void mover_bala(int ibala, int fil, int col);
```