UNIVERSIDAD DE MÁLAGA

ANDALUCÍA TECH
Campus de Excelencia Internacional

Dpto. de Lenguajes y Ciencias de la Computación

Programación de Sistemas y Concurrencia

Examen 1ª convocatoria ordinaria
Curso 2022-2023
30 de mayo de 2023

Bloque de Programación de Sistemas (3 puntos)
Descripción del Sistema

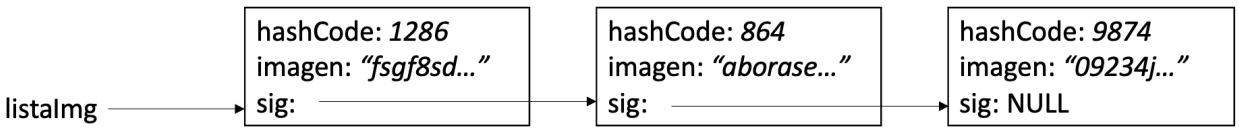
En este ejercicio se va a implementar una librería de gestión de imágenes de una cámara IP encargada de la detección de rostros. La librería permite **almacenar imágenes diferentes** (no duplicadas), lo hace en una lista. En esta lista se almacenan las imágenes **en orden de llegada**, es decir, que la primera imagen de la lista es la más antigua y la última la más nueva. La librería permite extraer las imágenes ya almacenadas. Para determinar si dos imágenes son iguales, cada imagen tiene un hash asociado de manera que si los hashes de dos imágenes coinciden entonces las dos imágenes son iguales.

Se han definido los siguientes tipos para implementar esta librería:

```
#define IMAGE_SIZE 64000                // Resolución MCGA (320×200).
typedef struct Imagen *ListaImagenes;
typedef struct Imagen
{
    int hashCode;                       // hash único para cada imagen
    char imagen[IMAGE_SIZE];            // datos de la imagen
    ListaImagenes sig;                  // puntero a la siguiente imagen
} Imagen;
```

Observa que una imagen almacena su código hash, los datos de imagen como una cadena de IMAGE_SIZE bytes, y un puntero a la siguiente imagen almacenada.

La siguiente figura muestra de forma visual la estructura usada. Para obtener la lista de imágenes de la figura se ha insertado primero la imagen con hashCode 1286, posteriormente se ha insertado la imagen con hashCode 864 y finalmente se ha insertado la imagen con hashCode 9874:



Para desarrollar este ejercicio se proporciona el archivo de cabecera *Imagenes.h* con las definiciones de tipos y las funciones. Además, se proporciona un archivo *main.c* con *asserts* para realizar algunas pruebas básicas de funcionamiento de la librería y un fichero binario *imagenes.bin* para cargar la información. Se deben implementar las siguientes funciones en el archivo *Imagenes.c*:

NOTA 1: Las puntuaciones de todas las funciones suman 10 puntos para facilitar la corrección, aunque posteriormente se prorrateará a 3 puntos, que es el valor de este bloque.
NOTA 2: Al final del enunciado se incluye un anexo con las funciones para manejo de ficheros y de cadenas de caracteres.

```
/**
 * @brief Inicializa la \p lista de imágenes vacía (0.25 pts.)
 * @param lista Lista de entrada para ser inicializada.
 */
void inicializarListaImagenes(ListaImagenes *lista);

/**
 * @brief Se inserta una nueva imagen según orden de llegada, ¡si no está!, con el \p
hascode y \p datos dados (1.75 pts). En caso de fallo al solicitar memoria, se sale del
programa mostrando antes un mensaje de error.
 * @param lista Lista de imágenes ya existente en el sistema.
 * @param hashcode Identificador único de la imagen.
 * @param datos Datos de la imagen, se deben copiar (incluye el carácter terminador)
* @return devuelve 0 si se puede insertar, 1 si no se puede insertar (duplicada).
 */
int insertarimagen(ListaImagenes *lista, int hashcode, char *datos);

/**
 * @brief Muestra las \p num primeras imágenes de la lista en orden de inserción
(primero la más antigua). Si hay menos de \p num, las mostrará todas. Para cada imagen
se debe mostrar el hascode y sólo los primeros 10 caracteres de sus datos (1 pt.).
 * @param lista Lista de imágenes ya existente en el sistema.
 * @param num Número máximo de imágenes a mostrar.
 */
void mostrarImagenes(ListaImagenes lista,int num);

/**
 * @brief Libera toda la memoria y deja la \p lista de imágenes vacía (1 pt.).
 * @param lista Lista de imágenes ya existente en el sistema.
 */
void destruirImagenes(ListaImagenes *lista);

/**
 * @brief Extrae la imagen más antigua y la quita de \p lista (0.5 pt.).
 * @param lista Lista de imágenes ya existente en el sistema.
 * @return Un puntero a la imagen más antigua o NULL si no existe.
 */
Imagen* extraercabeza(ListaImagenes *lista);

/**
 * @brief Extrae una ListaImagenes que contiene las \p num imágenes más antiguas y las
quita de \p lista (1.75 pt.).
 * @param lista Lista de imágenes ya existente en el sistema.
 * @param num Número mayor que cero que indica el máximo de imágenes a extraer en la
lista.
 * @return NULL si no hay ninguna imagen que extraer o una nueva lista con las, hasta
\p num, imágenes más antiguas.
*/
ListaImagenes extraerLista(ListaImagenes *lista, int num);
```

```
/**
 * @brief Escribe en un fichero de texto parte de los datos de las imágenes almacenadas
en la lista (1.75 pts.). El formato del fichero de texto será el siguiente: primero
tendrá una cabecera con una descripción de los campos. Tras esta cabecera, una línea
por cada imagen, ordenadas por antigüedad (las más antiguas primero). Solo se almacenan
los 100 primeros bytes de cada imagen. Ejemplo:

Hashcode;Imagen
1286; fsgf8sddsdfsdfnlnlsdfoisdfolskndflsndfoinlksngm ,m oiw09tuqargnl ... ;
864; aborasenhlakfhg,m ,miohlksndfgnqwoiut80q4760  oiqrnwglkjWR089TQH4OLGRN  ... ;
9874; 09234jfilnalkfngoiahrñktnmp9ur0q914u63091u3504ypihnñFGBP90rueyphnñRKG ... ;

 * @param filename Nombre del fichero para escribir en él.
 * @param lista Lista de imágenes del sistema.
 */
void guardarRegistroImagenes(char *filename, ListaImagenes lista);

/**
 * @brief Lee de fichero binario los datos de imágenes y los carga para su uso. En caso
de fallo al solicitar memoria, se sale del programa mostrando antes un mensaje de
error. La lista actual puede no estar vacía, recuerda antes borrar todas las imágenes
existentes (2.0 pts.).
 *  En el fichero binario se almacena la información de cada imagen con el siguiente
formato:
 *      un entero con el hascode de la imagen;
 *      una cadena de longitud IMAGE_SIZE bytes con la imagen. Esta cadena ya incluye
el carácter terminador '\0'.
 *  Se asume que las imágenes están guardadas por antigüedad, siendo las primeras las
más antiguas.
 * @param filename Nombre del fichero para leer de él.
 * @param lista Lista en la que se almacenan las imágenes leídas.
 */
void cargarRegistroImagenes(char *filename, ListaImagenes *lista);
```

Ejemplo de salida (los bytes de la imagen son aleatorios y pueden cambiar en cada ejecución):

```
Inicializado...
Insertamos una primera imagen
Insertamos una segunda
Insertamos una repetida
Mostramos todas las imagenes:
Mostramos imagenes
hascode 100 datos: qrjatydimo
hascode 200 datos: qrjatydpmo
hascode 300 datos: orjatydamp
hascode 400 datos: arjatwbamp
hascode 500 datos: dnjatwmamp
hascode 600 datos: vnjatwmamo
hascode 700 datos: dnjatamamo
hascode 800 datos: vnjatamano
hascode 900 datos: jnjataoano
hascode 1000 datos: knjataoaxo
Mostramos solo 5 imagenes:
```

Mostramos imagenes
hascode 100 datos: qrjatydimo
hascode 200 datos: qrjatydpmo
hascode 300 datos: orjatydamp
hascode 400 datos: arjatwbamp
hascode 500 datos: dnjatwmamp
Guardamos lista de imagenes:
Mostramos todas las imagenes:
Mostramos imagenes
hascode 200 datos: qrjatydpmo
hascode 300 datos: orjatydamp
hascode 400 datos: arjatwbamp
hascode 500 datos: dnjatwmamp
hascode 600 datos: vnjatwmamo
hascode 700 datos: dnjatamamo
hascode 800 datos: vnjatamano
hascode 900 datos: jnjataoano
hascode 1000 datos: knjataoaxo
Mostramos las 5 imagenes extraidas:
Mostramos imagenes
hascode 200 datos: qrjatydpmo
hascode 300 datos: orjatydamp
hascode 400 datos: arjatwbamp
hascode 500 datos: dnjatwmamp
hascode 600 datos: vnjatwmamo
Cargamos datos de binario
Mostramos lista de imagenes tras cargar de binario:
Mostramos imagenes
hascode 100 datos: hyxifwtdxj
hascode 200 datos: myxifwvwxj
hascode 300 datos: iyxifwvkrj
hascode 400 datos: uyxifwvyrj
hascode 500 datos: tyxifwvlrj
hascode 600 datos: ssxifwjkrj
hascode 700 datos: isxifwjkrj
hascode 800 datos: ksxifwjkrx
hascode 900 datos: isxifwjkry
hascode 1000 datos: esxtfwikry

ANEXO

MANIPULACIÓN DE CADENAS

Los prototipos de las funciones para manipular strings (incluidas en <string.h>) son:

char* strcpy(char *s1, char *s2): Copia los caracteres de la cadena s2 (hasta el carácter '\0', incluido) en la cadena s1. El valor devuelto es la cadena s1.

int strcmp(char *s1, char *s2): Devuelve 0 si las dos cadenas son iguales, <0 si s1 es menor que s2, y >0 si s1 es mayor que s2.

size_t strlen(const char *s): Calcula el número de caracteres de la cadena apuntada por s. *Esta función no cuenta el carácter '\0' que finaliza la cadena*.

MANIPULACIÓN DE FICHEROS

Los prototipos de las funciones para **manipulación de ficheros binarios** (incluidos en <stdio.h>) son los siguientes (se dan por conocidos los prototipos de las funciones de <stdlib.h> que necesites, como free o malloc):

FILE *fopen(const char *path, const char *mode): Abre el fichero especificado en el modo indicado ("rb"/"wb" para lectura/escritura binaria y "rt"/"wt" para lectura/escritura de texto). Devuelve un puntero al manejador del fichero en caso de éxito y NULL en caso de error.

int fclose(FILE *fp): Guarda el contenido del buffer y cierra el fichero especificado. Devuelve 0 en caso de éxito y -1 en caso de error.

LECTURA/ ESCRITURA TEXTO

int fscanf(FILE *stream, const char *format, ...): Lee del fichero *stream* los datos con el formato especificado en el parámetro *format*, el resto de parámetros son las variables en las que se almacenan los datos leídos en el formato correspondiente. La función devuelve el número de variables que se han leído con éxito.

int fprintf(FILE *stream, const char *format, …): Escribe en el fichero *stream* los datos con el formato
especificado en el parámetro *format*. El resto de parámetros son las variables en las que se almacenan los datos que hay que escribir. La función devuelve el número de variables que se han escrito con éxito.

LECTURA/ ESCRITURA BINARIA

unsigned fread(void *ptr, unsigned size, unsigned nmemb, FILE *stream): Lee *nmemb* datos, cada uno de tamaño *size* bytes, del fichero *stream* y los almacena en la dirección apuntada por *ptr*. Devuelve el número de elementos leídos.
unsigned fwrite(const void *ptr, unsigned size, unsigned nmemb, FILE *stream): Escribe *nmemb* datos, cada uno de tamaño *size* bytes, en el fichero *stream*. Los datos se obtenien desde la dirección apuntada por *ptr*. Devuelve el número de elementos escritos.