PRÁCTICA 2: Lectura de ficheros y uso de listas

OBJETIVOS

El objetivo de esta práctica es implementar un TAD Base de Datos que ofrezca una serie de servicios básicos para gestionar la base de datos de tweets y twitteros. Para ello se implementarán funciones de lectura de archivos que utilizarán los TAD desarrollados en la práctica anterior para mantener los datos leídos en memoria. Asimismo, se implementarán funciones de consulta.

Además de la documentación que los entornos de desarrollo proporcionan sobre el lenguaje de programación C, mediante la página de Moodle de la asignatura los profesores de los laboratorios proporcionarán las indicaciones adicionales para desarrollar el trabajo de cada una de las sesiones.

NORMATIVA DE ENTREGA

Como resultado de esta práctica debe entregarse mediante la herramienta Moodle, en la actividad que los profesores de los laboratorios indiquen, un fichero empaquetado (.zip) que contenga todos los ficheros que permitan abrir, construir el ejecutable y ejecutar su proyecto **antes de la primera sesión de la práctica siguiente. Cada grupo puede entregar hasta las 12 de la noche del día anterior al que su profesor iniciará la siguiente práctica.** Este fichero empaquetado debe, por tanto, contener al menos los siguientes ficheros:

- README.txt: fichero de texto plano donde han de constar vuestros datos (nombres, e-mail, grupo de prácticas, número de grupo) y cualquier otra indicación adicional sobre la práctica. Sobre la descripción de vuestro trabajo es suficiente con que expliquéis las cuestiones que os hayan podido parecer relevantes de vuestra solución. No se trata de que repitáis las instrucciones del enunciado ni las indicaciones que vuestros profesores hayan hecho en clase. Sólo debéis centraros en las peculiaridades que vuestra solución contenga. Añade en este fichero de forma clara la lista de comandos necesarios para generar los ejecutables.
- LISTA DE FICHEROS
    - twittero.h : Cabecera del TAD Twittero.
    - twittero.c : Implementación del TAD Twittero.
    - twitteros.h: Cabecera del TAD Twitteros.
    - twitteros.c: Implementación del TAD Twitteros.
    - tweet.h: Cabecera del TAD Tweets.
    - tweet.c: Implementación del TAD Tweets.
    - tweets.h: Cabecera del TAD Tweets.
    - tweets.c: Implementación del TAD Tweets.
    - lista.h
    - lista.c

- data_base.h: Cabecera del TAD base de datos
- data_base.c: Implementación del TAD base de datos.
- const.h: Fichero de cabecera con constantes globales de la aplicación.
- main2.c: Programa principal que incluye toda la funcionalidad de la práctica.

Se debe entregar todo en un zip.

El nombre del fichero comprimido tiene que seguir la siguiente estructura:

lab<turno>_gr<numero_grupo>_p2.zip

Donde:

- <turno> es el turno asignado (4111, 4112, 4113 o 4114)
- <numero_grupo> es el que identifica a tu grupo dentro de tu laboratorio

Es obligatorio respetar todos aquellos nombres y formatos que se describen en el enunciado.

¡No olvidéis incluir vuestros nombres en los ficheros entregados!

CRITERIOS BÁSICOS DE EVALUACIÓN

Las siguientes normas deben ser consideradas en la realización de los problemas propuestos y la entrega del material:

1. *Estilo de programación*: debe ajustarse a las normas básicas de estilo de la programación estructurada en C y que ya se han presentado en asignaturas anteriores de programación.
2. *Documentación del código*: todas las funciones que se programen deberán estar conveniente y brevemente comentadas, tanto en su cabecera (funcionalidad y parámetros de entrada/salida), como en su cuerpo (descripción de etapas relevantes). Se evitarán comentarios obvios o redundantes.
3. *Pruebas*: es especialmente importante el control de errores. Es justificable que un programa no admita valores muy grandes de los datos, pero no es justificable que el programa tenga un comportamiento anómalo con dichos valores.
4. *Comprobación y propagación de errores*: se deberán comprobar todos los errores susceptibles de producirse en el programa, incluyendo ausencia de archivo de datos de entrada, errores de formato en datos de entrada, fallo de funciones del sistema operativo (malloc, etc.). Se considerará fallo grave que el programa aborte su ejecución por un tratamiento de errores insuficiente o inadecuado.
5. *Liberación de memoria dinámica*: al finalizar los programas, éstos deberán haber liberado toda la memoria dinámica que hayan reservado durante su ejecución.
6. *Uso de Valgrind*. Es imprescindible que todos los ejercicios hayan sido probados usando Valgrind para asegurarse de que no existen problemas de memoria.

INTRODUCCIÓN

Para realizar la práctica se proporcionará una base de datos compuesta de dos ficheros que contienen twitteros y tweets. Estos ficheros siguen el formato que se describe a continuación.

- twitteros.txt
  Este fichero incluye la información relativa a los twitteros, con una línea por cada twittero de la base de datos. Los campos están separados por una barra ("|").

  - ALIAS: Nombre del twittero cuando elige su @alias.
  - NOMBRE: Nombre que muestra el twittero.

  E.g.:

```
user1|The special User-One
user2|User Master
user3|^_^
user4|Zascatrisqui
```

- tweets.txt
  Este fichero incluye la información relativa a los tweets, con una línea por cada tweet en la base de datos. Los campos están separados por una barra ("|") y son los siguientes:
  - ALIAS: alias del twittero que creó el tweet.
  - TWEET: Texto del tweet.

  E.g.:

```
user1|La #Vida es una #lenteja, @user2 @user3
user2|   . @user1 …o la tomas o la dejas!!!
user3|       ¿Eso no era de un #juego?
user4| La gente está muy mal, voy a hacer las prácticas de PROG2
```

1ª SEMANA: TAD Base de Datos - Funciones de lectura

Se implementará el TAD DataBase, que se encargará de la gestión. La estructura de datos y las operaciones se almacenarán en los ficheros data_base.h y data_base.c.
En esta primera semana se propone crear la estructura para almacenar las colecciones de twitteros y tweets, así como las funciones relativas a la lectura y escritura de los ficheros de datos.

- Crear la base de datos; la función toma como parámetro el directorio donde se encuentran los ficheros de datos de la base de datos.

```
data_base *new_data_base(char *dir_name);
```

- Eliminar la base de datos y liberar sus recursos:

```
void destroy_data_base(data_base *bd);
```

Se recomienda crear las siguientes funciones auxiliares para apoyar la lectura de los datos. Observa que estas funciones son auxiliares, es decir, su utilidad es facilitar la implementación del TAD pero su propósito no es que sean invocadas desde fuera del TAD. Por tanto, no deben aparecer en el fichero de cabecera.

- Función auxiliar para la lectura de los datos de un twittero. Esta función recibe un puntero a una cadena de caracteres y, comprueba que sigue el formato adecuado ALIAS|NOMBRE, devuelve un puntero a TAD twittero.
    - Es importante tener en cuenta que hay que eliminar el posible retorno de carro de la cadena. Debe decidirse si esta operación se realiza en esta función o si es responsabilidad del código que realiza la llamada a la función. En cualquier caso, lo importante es documentar la decisión en la documentación de la función.

```
twittero *bd_parse_twittero(char *str);
```

- Función auxiliar para leer un fichero con datos de twitteros. Crea la estructura *twitteros* y la rellena utilizando la información del fichero. Devuelve NULL si hay algún error al manejar el fichero.

```
twitteros *bd_parse_file_twitteros(char *file_name);
```

- Función auxiliar similar a la de lectura de twittero, pero para la lectura de los datos de un tweet. Esta función recibe un puntero a una cadena de caracteres, comprueba que sigue el formato adecuado *ALIAS|TWEET*, y devuelve un puntero al TAD *tweet*. Además recibe la lista de twitteros para enlazar cada tweet con su autor.

```
tweet *bd_parse_tweet(char *str, twitteros *aa);
```

- Función auxiliar para leer un fichero con datos de tweets. Crea la estructura *tweets* y la rellena utilizando la información del fichero. Devuelve NULL si hay algún error al manejar el fichero. Además recibe la lista de twitteros para enlazar cada tweet con su autor.

```
tweets *parse_file_tweets(char *file_name, twitteros *aa);
```

2 y 3ª SEMANA: TAD Base de Datos - Funciones de consulta

Se añadirán a la base de datos las siguientes funciones de consulta.

- Búsqueda de tweet y twittero según su identificador. Devuelve NULL si no se encuentra.

```
twittero *find_twittero_by_id(data_base *bd, long long int id);
tweet *find_tweet_by_id(data_base *bd, long long int id);
```

- Obtener todos los tweets y twitteros del sistema. El resultado será una lista (ver el TAD Lista proporcionado por los profesores).

```
list * find_all_twitteros(data_base *bd);
list * find_all_tweets(data_base *bd);
```

- Búsqueda de tweet y twittero por alias y nombre, respectivamente. Se utilizará la función *strstr* para realizar una búsqueda aproximada. El resultado será una lista que contendrá películas o actores según corresponda.

```
list * find_twittero_by_alias(data_base *bd, char *alias);
list * find_tweet_by_text(data_base *bd, char *text);
```

- Obtener todos los hashtags donde se menciona al twittero indicado por 'alias' (no los hashtags que escribe un twittero, sino aquellos hashtags en los tweets que se menciona a @alias). Devolverá una lista cuyo dato es una estructura nueva con el hashtag y su número de repeticiones [ver estructura 'hashtag_number'].

```
list *find_hashtags_per_mentioned(data_base *bd, char *alias);
```

4ª SEMANA: TAD Base de Datos – Menú del sistema

Algunas de las funciones de consulta realizadas en la semana anterior servirán para implementar las operaciones de inserción en la base de datos, que deben comprobar que la base de datos queda en un estado consistente. Estas operaciones se describen a continuación.

Insertar datos

- Dos funciones para añadir un twittero y un tweet respectivamente. La primera función debe comprobar que no exista en la base de datos un twittero con el mismo alias que el twittero pasado como parámetro.
  Adicionalmente, para los tweets, debe comprobarse que todos los usuarios mencionados en el tweet ya han sido añadidos a la base de datos anteriormente.
  Si no es posible añadir el elemento a la base de datos se devolverá FALSO.

```
BOOL data_base_add_twittero(data_base *bd, twittero *a);
BOOL data_base_add_tweet(data_base *bd, tweet *f);
```

Se modificará el menú creado en la práctica anterior, para considerar ahora que las colecciones de tweets y usuarios están mantenidas por el TAD Base de Datos, y ampliando el menú del sistema con nuevas opciones, como se muestra a continuación.

```
$ ./practica2 datos/

   ************************************
   *       1. Twittear                *
   *       2. Cuentas                 *
   *          2.1 Iniciar sesión      *
   *          2.2 Registrarse         *
   *       3. Consultas               *
   *          3.1 Todos los tweets    *
   *          3.2 Todos los twitteros *
   *          3.3 Buscar tweet por ID *
   *          3.4 Buscar twittero por ID *
   *          3.3 Buscar tweet por un texto   *
   *          3.4 Buscar twittero por alias *
   *          3.5 Mostrar hashtags para usuario mencionado *
   *       5. Salir                   *
   ************************************
   Elige una opción >
```

La ejecución de cada consulta mostrará el tiempo empleado en recuperar los datos (sin incluir el tiempo requerido para imprimir por pantalla). A continuación se muestra un ejemplo de cómo medir el tiempo de ejecución de una operación:

```
#include <stdio.h>
#include <sys/time.h>
int main(void)
{
    struct timeval ti, tf;
    double tiempo;
    gettimeofday(&ti, NULL);   // Instante inicial

    printf("Lee este mensaje y pulsa ENTER\n");
    getchar();

    gettimeofday(&tf, NULL);   // Instante final
    tiempo = (tf.tv_sec - ti.tv_sec)*1000 +
             (tf.tv_usec - ti.tv_usec)/1000.0;
    printf("Has tardado: %g milisegundos\n", tiempo);
}
```