ESTRUCTURA DE DATOS

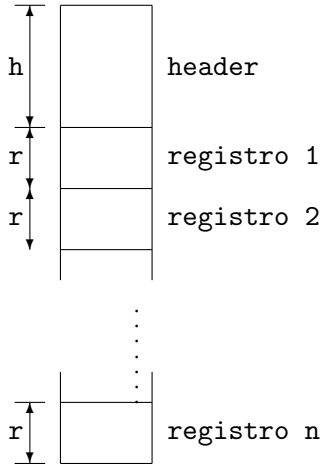
PRÁCTICA – FICHEROS

Calendario de entregas (de la primera parte)

grupo	comienzo	entrega

ENUNCIADO DE LA PRÁCTICA

En esta práctica se desarrollará una librería de funciones para gestionar ficheros con registros de longitud fija de una base de datos. La base de datos es un fichero binario que contiene una cabecera con información general seguida por los registos. SIendo los registros de longitud fija, no será necesario poner una indicación de la longitud del registro en cada registro: simplemente habrá un campo de la cabecera donde se pondrá la longitud de los registro. EL fichero, por tanto, tendrá una organización muy sencilla:



La cabecera contendrá información de carácter general sobre el fichero:

```
typedef struct tagBDCabecera {
        unsigned short int tamanioCabecera;  /* tamanio de Cabecera*/
        int lNumeroRegistros;   /* numero total de registros */
        int  tamanioRegistro;   /* tamanio de registro, 0 si tamanio variable */
        char pszFechaCreacion[9];   /* fecha de creacion de la BD - DD/MM/YY */
```

```
        char pszFechaActualizacion[9];  /* fecha de ultima modificacion de la BD - DD/MM/YY */
        char pszHoraActualizacion[9];   /* hora de utima modificacion de la BD - hh/mm/ss */
        long lPosicionBorrado;   /* posicion del registro borrado con mayor tamano */
        char sRelleno[128];  /*campo auxiliar */
} BDCabecera;
```

El campo `sRelleno`, de 128 carácteres, es a disposición para cualquier cosa que podáis necesitar. Veremos a continuación que con el mismo esquema se implementarán tres tipos de ficheros: un fichero de datos, y dos típos de índices: el campo `sRelleno` se usará en este caso para introducir un indicador del tipo de fichero a que se refiere la cabecera. Aquí también podéis insertar, si queréis, el nombre del autor del fichero.
Las funciones a implementar que permiten el manejo de la BD se dividen en dos tipos.

Manipulación básica

Funciones que permiten la manipulación básica del fichero de la base de datos (creación, lectura, inserción, borrado, actualización). Son independientes de los campos que componen el registro, siempre que el registro siga el formato básico que se indica a continuación. Estas funciones, junto con las estructuras de datos que utilizan, se encuentran especificadas en el fichero `bd.h`
Los registros son de longitud fija, por tanto cada registro (desde el punto de vista de estas funciones) es simplemente una cadena de *byte* de cierta longitud. Las funciones de la librería BD de bajo nivel *no se ocupan* del contenido de los registros: simplemente reciben en un buffer de memoria una cadena de la longitud oportuna y lo escriben en la posición oportuna del disco, o leen una cadena de byte de la longitud oportuna y la copian en el buffer de memoria.
Las funcionen añaden un byte al registro: el primer byte (invisible para las funciones de alto nivel) indica si el registro está disponible o si está borrado: si el registro está oculado, este byte contendrá el caracter '!' mientras si el registro está borrado, el *byte* contendrá el caracter '*'. Como hemos visto en clase, en un fichero de base de datos, los registros borrados no se borran de verdad: simplemente se marcan como "borrados" e se dejan en el fichero.
Atención: el byte que indica si el registro está borrano *no entra* en la cuenta de la longitud del registro. La longitud del registro la declara (en el momento de la creacción de la base de datos) el programa que usa el fichero, y este programa no sabe nada de registros borrados ni de marcas. Por tanto, si el programa principal quiere una base de datos con registros de n *byte*, internamente se crearán y usarán registros de $n+1$ *byte*. En este sentido podría ponerse un problema: ¿Que valor almacenamos en la cabecera, n o $n+1$? Dado que el programa principal no usa la cabecera (la información sobre el fichero se escribe y se lee sólo usando funciones oportunas de la librería BD) la cosa es indiferente, y podéis elegir lo que más os parece conveniente. Lo que sí es importante es que cuando el programa principal quiera saber el número de *byte* de los registros, el valor que reciba sea n, y no $n+1$.
Cuando se quiere insertar un registro en la base de datos, será necesario primero buscar si hay registros borrados disponibles. Si los hay, se insertará el nuevo registro en el primer registro borrado disponible, si no se añadirá el registro al final del fichero.

Funciones especificas

Funciones que permiten operar con una base de datos específica. En esta primera parte de la práctica, manejaremos una base de datos de películas, sacándola del fichero de texto proporcionado con la práctica. El fichero de texto tiene el formato siguiente:

```
id  title                 year    name                role    character
24485 'Breaker' Morant (1980) 1980 Donovan, Terence (I) actor Capt. Simon Hunt
59632 'Breaker' Morant (1980) 1980 Fitz-Gerald, Lewis actor Lt. George Ramsdale Witton, Bushveldt Carbineers
```

```
100791 'Breaker' Morant (1980) 1980 Gray, Ian (I)         actor B/M Thomas
  .
  .
  .
```

Nótese que los campos *role* y *character* pueden contener espacios, pero su terminación e reconocible en cuanto *role* se separa del campo siguiente con un tabulador (caracter `\t`) y al campo carácter termina cuando termina la línea (caracter `\n`). También hay que notr que casi todos los campos son de longitud variable. Con el fin de almacenar estos datos en un fichero con registros de longitud fija será necesario establecer una longitud máxima y cortar los campo cuya longitud sobrepasa la establecida.

1 EJERCICIOS A REALIZAR

Cada vez que se realice una modificación en la base de datos (se añada, borre o actualice un registro), se almacenará la fecha y hora de actualización en la estructura de la cabecera del fichero de la BD. Se escribirá la cabecera en el disco justo antes de cerrar la BD. La inserción y actualización en presencia de registros borrados deberá realizarse según la técnica de inserción worst-fit.

i) Utilizando las funciones especificadas en los ficheros `bd.h` escribir un programa llamado `creabd` que genere la base de datos binarias de películas con el formato descrito arriba a partir de un fichero de texto de películas (`peliculas.txt`). Este fichero de texto recoge los datos de las películas (una por cada línea) cuyos campos están separados por *uno o más espacios*. El programa `creabd` recibirá como parámetros el nombre del fichero de texto y el nombre de la base de datos binaria a crear (p.e. `pelis.dat`). Un ejemplo de llamada a este programa es:

```
creabd peliculas.txt pelis.dat
```

ii) Para facilitar la corrección y el depurado de las funciones implementadas en el ejercicio anterior, se creará un programa ejecutable llamado `visualiza`. Este programa actuará sobre una base de datos ya creada en el formato especificado y presentará por pantalla el contenido de la cabecera y los registros en modo texto. Deberá aceptar como parámetro el nombre de la base de datos (`ciudades.dat`). El programa deberá mostrar la siguiente información:

- **a.** Al comienzo, la información contenida en la cabecera de la base de datos.
- **b.** En caso de registros no borrados, el programa mostrará en una línea: el contenido de cada registro y su tamaño total.
- **c.** En caso de registros borrados, el programa imprimirá una línea que contendrá tres asteriscos.
- **d.** Al final, imprimirá un resumen con el número de registros borrados y el tamaño total de los registros borrados (fragmentos externos).

Obs: Se deberá probar el funcionamiento de este programa después de borrar registros de la BD, para comprobar que el cálculo de la fragmentación externa esté debidamente realizado.

iii) Escribir el programa `borra` para eliminar las peliculas producidas antes de un año dado. Al borrar un registro, se insertará una marca de borrado en el primer *byte* del registro la marca de borrado. Además se deberá actualizar el campo `lNumeroActivos` de la cabecera. El programa recibirá como parámetros de entrada la base de datos binaria y la el nńero límite de habitantes. Ejemplo:

```
borra ciudades.dat 1970
```

(elimina las películas con fecha inferior a 1970).

iv) Escribir el programa `inserta` que añade a una base de datos existentes las películas contenidas en un un fichero de texto. EL programa insertará las películas utilizando por primera cosa los registros borrado: por cada inserción, si el número total de registros es superior al número de registros activos, habrá que buscar un registro borrado en el fichero y almacenar el nuevo registro en el lugar del registro borrado. Si no hay registros borrados, el nuevo registro se insertará al final del fichero. Por ejemplo, la llamada

```
inserta ciudades.dat mas_peli.txt
```

inserta en la base de datos `ciudades.dat` todas las pelis contenidas en el fichero `mas_peli.txt`

Consejos: Como ya deberíais saber de otras asignaturas, es buena práctica de programación crear módulos (ficheros ".c") que reunen las funciones que trabajan sobre una cierta estructura o que operan sobre un tipo de datos dado. En ningún caso se deberán crear funciones `main` muy complejas, ni poner funciones de procesamiento en el fichero `main.c`. La función `main` deberá limitarse a leer los parametros de la línea de comando, verificar que los valores sean válidos, quizás abrir algún fichero, y llamar las funciones de procesamiento oportunas. En este caso, la única indicación fija que se da es relativa a las funciones de bajo nivel, declaradas en el fichero `bd.h`. Habrá, sin duda, otro módulo (como mínimo) dedicado a la gestión de la estructura *pelicula*.
Cabe recordar (otra vez) que las funciones de `bd.h` no se ocupan de los datos que se almacenan en el fichero: para estas funciones cada registro es una cadena de *byte* que se devuelven con la función `BDData`. Habrá que crear una estructura especifica para peliculas, algo como

```
typedef struct {
   int id;
   char *title;
   int year;
   char *name;
   char *role;
   char *character;
} peliculas;
```

y crear funciones para leer estas estructuras del fichero ".txt", transformar una estructura en una cadena de *byte* y transformar una cadena de *byte* en una estructura (esta es una parte muy divertida donde se hacen cosas bonita y un poco friqui con los punteros... la veremos en clase). (También recordaos que una cadena en C es terminada con un `\0`, por tanto si decidimos que el título lo cortamos a los 20 carácteres y dejamos 20 *byte* en el fichero no cabe el `\0` final y por tanto lo que se almacena no es una cadena...¿Cómo lo resolvemos?)

¡Buen trabajo!

Nota: Recordad que debéis emplear un único fichero Makefile para generar los ejecutables.