APRENDERAPROGRAMAR.COM

LEER Y ESCRIBIR ARCHIVOS O FICHEROS EN C. FOPEN, FCLOSE, MODOS DE ACCESO READ, WRITE Y APPEND (CU00536F)

Sección: Cursos
Categoría: Curso básico de programación en lenguaje C desde cero

Fecha revisión: 2031

Resumen: Entrega nº36 del curso básico "Programación C desde cero".

Autor: Mario Rodríguez Rancel

LEER Y ESCRIBIR A ARCHIVOS EN C

Advertencia: una manipulación incorrecta de ficheros puede dar lugar a la pérdida de datos, al malfuncionamiento de programas o a fallos del sistema operativo del ordenador. Es necesario disponer de copias de seguridad de toda información que se considere importante.

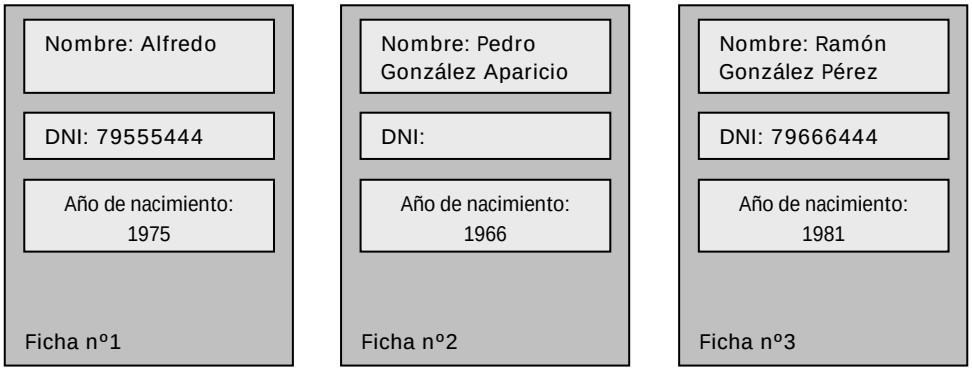
Guardar datos a un disco o recuperar los datos previamente guardados son procesos fundamentales en cualquier programa informático. La importancia del almacenamiento de contenidos es obvia: envío de información a otros usuarios, posponer el trabajo varios días o semanas sin tener que introducir manualmente los datos de nuevo, acceso a información almacenada en sistemas remotos, etc. Incluso para desarrollos de software de relativamente corta longitud resulta relevante la gestión de datos, por ahorrar una cantidad de tiempo considerable.

Ficheros y bases de datos constituyen un cuerpo de conocimiento de cierta complejidad y extensión. Por motivos de espacio no vamos a hacer más que una breve introducción.

Podemos pensar en los ficheros o archivos de forma análoga a lo que sería un archivo físico: un lugar donde hay información almacenada. Ahora bien, eso no nos dice cómo se encuentra esa información: si está ordenada (alfabéticamente, de mayor a menor, en impresos con campos predefinidos), si son datos de un tipo o de varios tipos (numéricos, tipo texto, mezcla de tipos de datos...), si se leen usando alguna clave de interpretación (una partitura no se lee igual que un texto), etc. Todo esto es relevante y por ello normalmente sabremos qué tipo de datos contiene el fichero y cómo están ordenados antes de acceder a él. No es lo mismo guardar "*3246*" como número tipo *int* que como cadena de caracteres. Un tipo *int* ocupa un espacio en disco, p.ej. 2 *bytes* y tiene una "clave" para su lectura, mientras que como caracteres cada cifra ocupa un *byte* y se lee como si de texto se tratara.

Un asunto a considerar es si la información está contenida en campos de longitud predefinida. Consideremos un archivo donde se almacenan los nombres, números de DNI y año de nacimiento de 3 personas. Podemos considerar diferentes casos:

a) Existen campos de longitud predefinida, que pueden carecer de información por no estar disponible.



En el fichero físico la información la guardamos en hojas preimpresas donde cada apartado tiene una longitud disponible para escribir en él. En el fichero informático la información la guardamos en registros (un registro equivale a una hoja del fichero físico), constando cada registro de 3 campos (campo "Nombre", campo "DNI" y campo "Año de nacimiento"). Los campos serían por ejemplo:

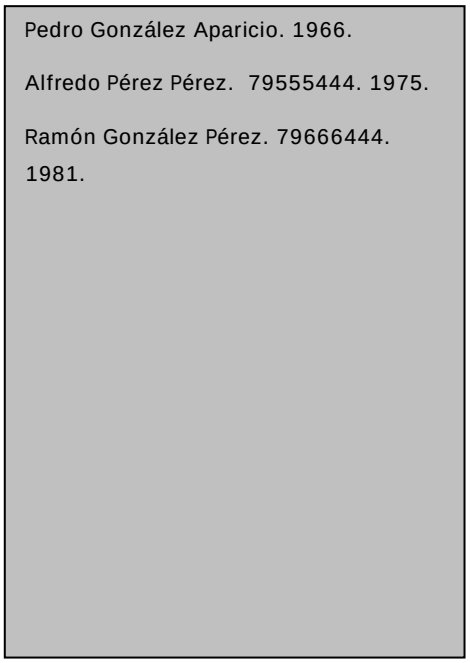
Nombre: tipo texto 25 caracteres de longitud.

DNI: tipo entero largo (*long*).

Año de nacimiento: tipo entero (*int*).

Supongamos que cada registro ocupa *25 + 4 + 2 = 31 bytes*. Si tenemos tres registros el fichero ocupa *31 * 3 = 93 bytes*, independientemente de que los datos realmente existan o no, ya que el espacio se encuentra disponible para albergar un dato.

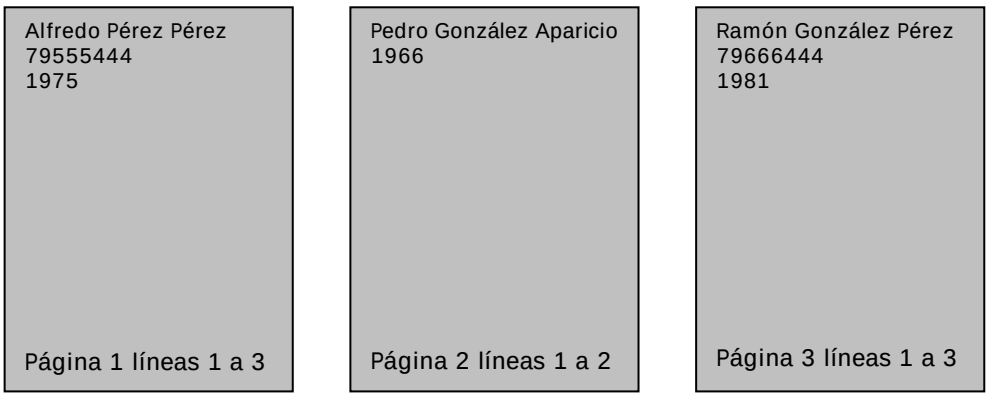
b) La información no está en campos de longitud predefinida.



En el fichero físico la información la guardamos en hojas donde escribimos los datos disponibles para cada persona. En el fichero informático la información la guardamos por ejemplo en líneas (cada línea equivale a la información disponible de una persona) sabiendo que cada línea puede tener información de uno o varios aspectos (nombre, DNI y edad) y una longitud variable.

No sabemos cuántos *bytes* nos ocupa cada línea ni cuántos *bytes* nos ocupa el fichero. No sabemos en qué posición se encuentra una información dada porque no tenemos referencias de situación. Ante ello, optamos por hacer una lectura tipo "texto" en la que un número de 8 cifras nos va a requerir *8 bytes* frente a los *4 bytes* que supondrían como tipo *long.*

c) La información no está en campos de longitud predefinida pero se dispone de información referente a su ubicación.



En el fichero físico la información la guardamos en hojas indexadas, donde cada dato dispone de un localizador. Por ejemplo, en la página 3 línea 3 encontramos el dato correspondiente al año de nacimiento de Ramón González. En el fichero informático la información la guardamos por *bytes*, sabiendo que cada tipo de dato ocupa un número de *bytes* determinado. Así si guardamos *Importe: 3235*, podríamos hacerlo ocupando *11 bytes* desglosados en *9 bytes* (uno por carácter, incluidos los espacios) asociados a "Importe: " como *tipo texto* + *2 bytes* (para un tipo *int*) asociados al número *3235*.

También podríamos guardarlo todo como tipo texto ocupando *13 bytes*.

Una vez guardada la información, podríamos recuperarla indicándole al ordenador: *Extrae como tipo texto los bytes comprendidos entre 1 y 9 y como número los bytes comprendidos entre 10 y 11.*

Hemos tratado de introducir, a través de la analogía con un fichero físico, enfoques básicos para la comprensión y gestión de ficheros informáticos. La problemática, como se comprenderá, es muy amplia al intervenir numerosos factores (hardware y software; directorios, unidades de disco, sobreescritura de archivos, borrado accidental de datos, etc.) y posibilidades (datos ordenados, parcialmente ordenados o desordenados, diversidad de tipos de datos, tramos de ficheros con registros y campos, etc.).

Vamos a ver simplificadamente cómo podemos gestionar ficheros con el lenguaje C.

ACCESO A FICHEROS CON C: SECUENCIAL, ALEATORIO O BINARIO.

C nos permite el acceso a ficheros gracias a la biblioteca studio.h, que nos facilita todo lo necesario para el trabajo con ficheros. El acceso a un fichero puede ser:

a) ***Acceso secuencial:*** orientado a su uso en archivos donde la información está dispuesta como tipo texto.
b) ***Acceso aleatorio:*** orientado a su uso en archivos con datos contenidos en registros de longitud fija, que a su vez pueden estar subdivididos en campos. Un campo puede contener un valor numérico o ser tipo texto.
c) ***Acceso binario:*** permite guardar datos con el orden y estructura que se desee, siendo posible acceder a ellos conociendo la posición (número de *byte*) en que se encuentran.

Al existir distintas formas de acceso, se habla de "archivos secuenciales", "archivos aleatorios" y "archivos binarios", aunque esta terminología debe tomarse con precaución. Es decir, un "archivo binario" hace referencia a un archivo donde habitualmente guardamos y extraemos datos usando el acceso binario. Sin embargo, podríamos usar este archivo con otro tipo de acceso. En resumen, un archivo es "información" y la información puede reordenarse, cambiar, manipularse de distintas maneras, etc. Usar un tipo de acceso en un momento dado no obliga a usarlo siempre, aunque así pueda ser.

En la analogía fichero físico - fichero informático, las equivalencias aproximadas serían las de:

- Archivo secuencial = La información no está en campos de longitud predefinida.

- Archivo aleatorio = Existen campos de longitud predefinida, que pueden carecer de información por no estar disponible.

- Archivo binario = La información no está en campos de longitud predefinida pero se dispone de información referente a su ubicación.

Repetimos que todo es lo suficientemente complejo como para que sea difícil dar definiciones exactas. Nosotros vamos a estudiar únicamente ciertas formas de extraer y guardar datos de un fichero secuencial a través de algunas funciones disponibles en el lenguaje para este fin. Esto será suficiente para poder crear programas que leen y guardan datos desde archivos de forma simple. El resto de instrucciones y posibilidades de acceso no será objeto de estudio ya que quedan fuera de nuestros objetivos.

ABRIR Y CERRAR FICHEROS SECUENCIALES EN C. FILE, FOPEN, FCLOSE.

Para manipular información de un fichero, por ejemplo guardar datos en él o leer datos desde él, lo primero que hemos de hacer es abrir el fichero. El proceso a seguir es:

Abrir el fichero → Manipular datos (extraer, guardar, modificar, etc.) → Cerrar el fichero

fopen → Manipular datos → fclose

A su vez, cada vez que accedemos a un fichero hemos de indicar “bajo qué modalidad” vamos a utilizar ese fichero: si es para leer datos, para escribir datos, y a su vez si se trata de escribir si debemos añadir los datos a los que ya existieran en el fichero o si debemos desechar lo existente y crear el fichero como si fuera nuevo.

La sintaxis que emplearemos para escribir o leer datos en ficheros secuenciales será la siguiente:

```
FILE* nombreInternoFichero;
      nombreInternofichero = fopen (“rutaNombreFichero.txt”, “modoDeAcceso”);
      operación 1; operación 2; … ; operación n;
      …
fclose (nombreInternofichero)
```

FILE* nombreInternoFichero: para manipular un fichero necesitamos "algo" que el programa utilizará para almacenar información sobre el fichero (como el modo de acceso, ruta, etc.). Ese "algo" (estructura de datos) lo creamos escribiendo FILE* y el nombre que queramos utilizar para referirnos a la estructura de datos que almacenará la información relacionada con el fichero. Cuando creamos esta estructura de datos (en realidad deberíamos usar el término puntero, pero vamos a mantener las ideas claras usando simplificaciones) la creamos vacía (su contenido es NULL), a la espera de rellenarla con información.

nombreInternoFichero = fopen ("nombreFichero.txt", "modoDeAcceso"): es la instrucción que da lugar a que la estructura de datos deje de estar vacía y comience a contener información y a ser útil para poder actuar sobre un fichero. Para aportar la información necesaria se usa la función *fopen*, que abre el fichero cuya ruta y nombre se especifica en *rutaNombreFichero.txt* (no necesariamente tiene que ser un fichero txt, pero de momento vamos a suponer que es así) y define el modo de acceso. El modo de acceso indica para qué se abre y cómo debe tratarse el fichero según si existía previamente o si no existía.

Operación 1; Operación 2; ...: indica las operaciones que se ejecutan. Pueden ser bucles, condicionales, o instrucciones simples que normalmente se encargarán de extraer información del fichero (leer datos) o incorporar información al fichero (escribir datos).

fclose (nombreInternoFichero): da lugar a la conclusión de procesos abiertos, al cierre del fichero y a que "la línea de comunicación" quede libre para una posterior ocasión. Siempre que abramos un fichero con *fopen* deberemos cerrarlo con *fclose*.

MODOS DE ACCESO O APERTURA DE ARCHIVOS CON C. LEER (READ), ESCRIBIR (WRITE) O AÑADIR (APPEND).

Para acceder a un fichero se escriben sentencias similares a estas:

```
FILE* fichero; fichero = fopen("cursoAF1.txt", "wt");
```

wt es una clave identificadora que consta de dos componentes: w y t. Vamos a definir cuál es el significado de las dos componentes de "wt" y qué valores pueden aparecer en este lugar y su significado. La comprensión se facilita si tenemos en cuenta que r proviene de *read*, w proviene de *write* y a proviene de *append*.

Primer componente (modo)	Significado
r	Abre un archivo existente para lectura.
w	Crea un nuevo archivo para escritura (si ya existe, se pierden los datos preexistentes).
a	Abre el archivo para añadir datos al final (se conservan datos existentes), o crea el archivo si no existía.

El primer componente puede ir acompañado de un símbolo +, por ejemplo r+. El símbolo + indica que se permite la lectura y la escritura. Por ejemplo r+ indica apertura de un archivo existente para lectura o escritura, w+ indica creación de un nuevo archivo para lectura o escritura y a+ indica apertura de un archivo para lectura o escritura al final.

El segundo componente indica el tipo de archivo que se abre.

Segundo componente (tipo de archivo)	Significado
t	Archivo de texto
b	Archivo binario

EJERCICIO

En un programa escrito en C nos hemos encontrado esta línea:

```
fichero = fopen("cursoAF1.txt", "wt");
```

En otro programa nos hemos encontrado esta otra línea:

```
fichero = fopen("cursoAF2.txt", "rt");
```

Responde a estas preguntas:

a) ¿Manejan ambos programas el mismo tipo de archivo? ¿Por qué?

b) ¿Qué ocurrirá en cada programa si el archivo no existe?

c) ¿Con qué fin se abren los archivos en cada programa?

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU00537F

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:
http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=82&Itemid=210