



**UCAM**  
UNIVERSIDAD  
CATÓLICA DE MURCIA

2020

GRUPO **\$** **TUDIUM**  
FORMACIÓN



# Tema 1

FICHEROS

STUDIUM

[www.grupostudium.com](http://www.grupostudium.com)  
[informacion@grupostudium.com](mailto:informacion@grupostudium.com)  
954 539 952

## Introducción

Al aprender cualquier lenguaje de programación, uno de los primeros conceptos con los que se trabaja son las **variables**, que nos sirven para guardar información. Pero esa información es **volátil**, es decir, en cuanto se apague el ordenador, se pierde. Para poder **persistir** la información más allá de la actual ejecución de un programa necesitamos guardarla en un medio **no volátil**, como puede ser el disco duro del sistema. Aprenderemos en este tema a guardar los datos de nuestros programas en archivos o ficheros para su posterior uso.

Aprenderemos a usar la **librería** que trae el lenguaje C para trabajar con ficheros, veremos las principales **operaciones** que podemos realizar con los datos de los ficheros y aprenderemos a diferenciar los **ficheros de texto** de los **ficheros binarios**.

## Librería estándar de C

La librería estándar de C **stdio.h** es la que nos va a permitir el trabajo con ficheros, de manera, que, en todos nuestros ejemplos y ejercicios, debemos incluir esta librería para poder realizar el trabajo correctamente.

## Operaciones básicas con ficheros: Apertura, lectura, escritura, renombrado, borrado y cierre

Cuando se trabaja con ficheros, debemos realizar una serie de operaciones antes de disponer de la información de estos, o antes de meterles información. Igualmente hay que realizar una operación de cierre al finalizar el trabajo con los mismos.

### Apertura

Para trabajar con un fichero, debemos, en primer lugar, abrirlo. Esto implica varios aspectos: debemos indicar la **ubicación** del archivo, el **nombre** y **para qué** lo queremos abrir. Esta operación supone la creación de un **vínculo** entre el fichero físico guardado en disco duro con nuestro programa, para que la información fluya entre ellos.

### Operaciones

Una vez abierto el archivo, podemos trabajar con él. Podemos recorrerlo para sacar su información, podemos incluir nueva información respetando la que haya o machando el contenido, etcétera. Dependiendo del **contenido** de los ficheros, tendremos que usar unas funciones u otras, como veremos más adelante.

### Cierre

Una vez acabado el trabajo con cualquier archivo, es altamente recomendable que se cierre la conexión, es decir, que se destruya el vínculo creado en la apertura. Esto

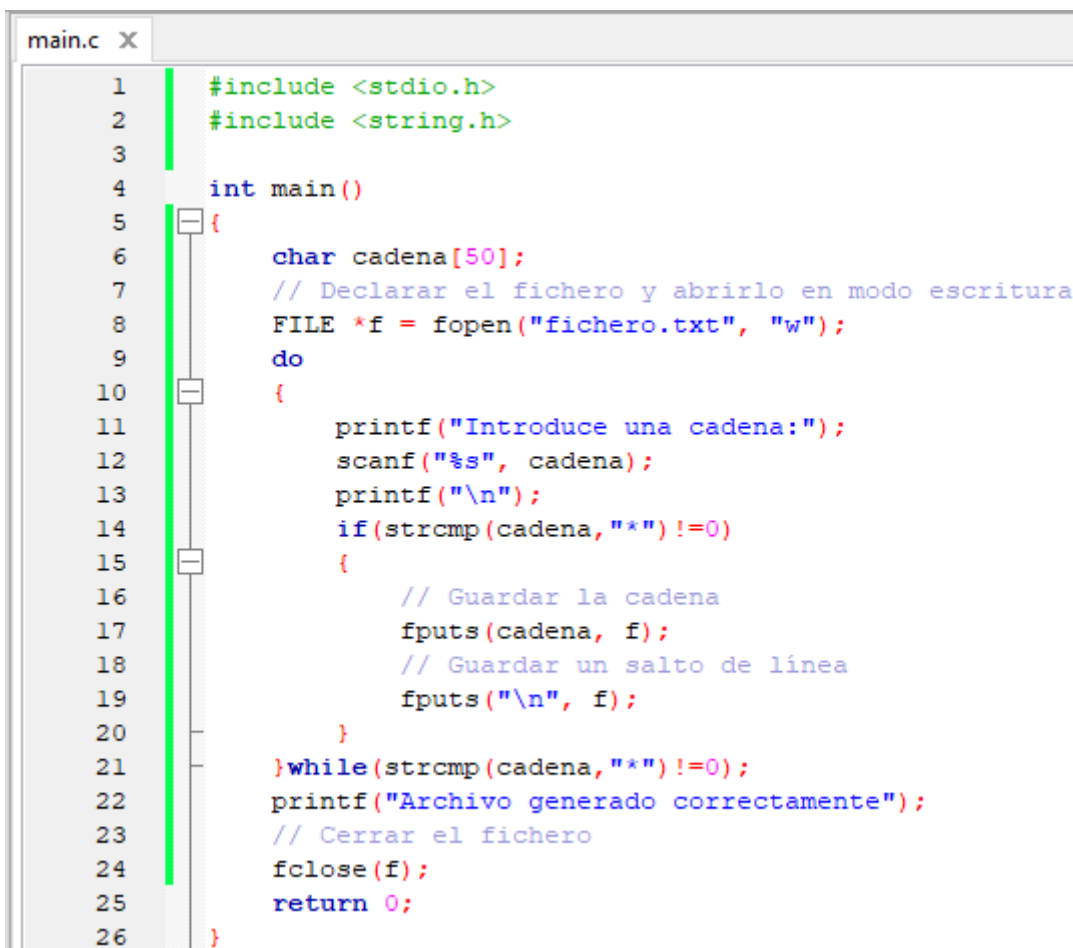
evita una posible pérdida de información en el fichero y también supone dejar “libre” el fichero para que otro programa o proceso puede interactuar con él.

## Ficheros de texto

Cuando trabajamos con ficheros cuyo contenido es **texto plano**, como puede ser un fichero txt, un html o un css usaremos las siguientes funciones para trabajar con ellos:

- **fopen()** para abrir el fichero
- **fgets()** para sacar texto del fichero
- **fputs()** para meter texto en el fichero
- **fclose()** para cerrar el fichero

Veamos con ejemplos cómo trabajar con ficheros de texto. Realizaremos un programa que irá leyendo palabras por teclado hasta que una de ellas sea un asterisco, y las irá guardando en un fichero llamado “fichero.txt”:



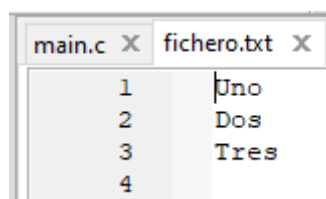
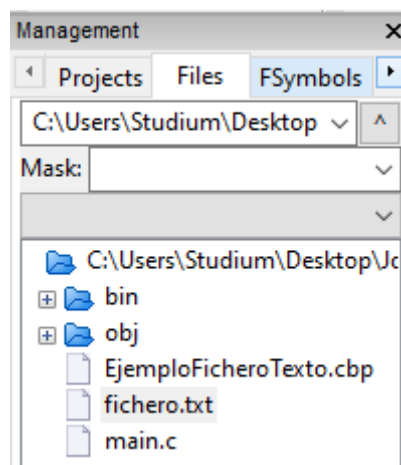
```
main.c X
1  #include <stdio.h>
2  #include <string.h>
3
4  int main()
5  {
6      char cadena[50];
7      // Declarar el fichero y abrirlo en modo escritura
8      FILE *f = fopen("fichero.txt", "w");
9      do
10     {
11         printf("Introduce una cadena:");
12         scanf("%s", cadena);
13         printf("\n");
14         if(strcmp(cadena, "*") != 0)
15         {
16             // Guardar la cadena
17             fputs(cadena, f);
18             // Guardar un salto de línea
19             fputs("\n", f);
20         }
21     }while(strcmp(cadena, "*") != 0);
22     printf("Archivo generado correctamente");
23     // Cerrar el fichero
24     fclose(f);
25     return 0;
26 }
```

```
"C:\Users\Studium\Desktop\Jorge\UCAM\Fundame..."
Introduce una cadena:Uno
Introduce una cadena:Dos
Introduce una cadena:Tres
Introduce una cadena:*
Archivo generado correctamente
Process returned 0 (0x0)   execution time : 12.952 s
Press any key to continue.
```

## NOTAS:

- **Línea 8:** Declaramos una variable de tipo **FILE** que llamamos **f**, abrimos el fichero de nombre **fichero.txt** con **fopen()** en modo **escritura**. Los modos son:
  - **w** para indicar escritura (write) machacando contenido anterior
  - **r** para indicar lectura (read)
  - **a** para indicar escritura (add), pero respetando el contenido previo
- Esta variable **f** debe ser un **puntero**, por lo que se le antepone el asterisco (\*).
- **Línea 17:** Con la función **fputs()** metemos la cadena en el fichero **f**.
- **Línea 24:** Cerramos el fichero con **fclose()**.

Desde el propio **Code::Blocks** podemos abrir el fichero de texto recién creado para ser su contenido. En la barra de herramientas izquierda, accedemos a la pestaña "Files" que nos muestra el proyecto desde el punto de vista de los ficheros del sistema operativo que lo conforman:



Ahora, vamos a realizar un programa que lea el contenido del fichero y lo muestre por pantalla:

```
main.c X
1  #include <stdio.h>
2
3  int main()
4  {
5      char cadena[50];
6      // Declarar el fichero y abrirlo en modo lectura
7      FILE *f = fopen("fichero.txt", "r");
8      while (fgets(cadena, 50, f) != NULL)
9      {
10         printf("%s", cadena);
11     }
12     // Cerrar el fichero
13     fclose(f);
14     return 0;
15 }
```

```
"C:\Users\Studium\Desktop\Jorge\UCAM\Fundam...  -  □  X
Uno
Dos
Tres

Process returned 0 (0x0)   execution time : 0.044 s
Press any key to continue.
```

#### NOTAS:

- **Línea 7:** Declaramos una variable de tipo **FILE** que llamamos **f**, abrimos el fichero de nombre **fichero.txt** con **fopen()** en modo **lectura**.
- **Línea 8:** Mediante la instrucción **fgets()** leemos la información del fichero, línea a línea. Cuando ya no haya más información, **fgets()** devuelve NULO y se acaba el bucle. La función **fgets()** tiene tres parámetros:
  - El primero es la cadena donde se guarda lo que se lee.
  - El segundo es el tamaño de la cadena anterior
  - El último es el fichero de donde sacar la información
- **Línea 13:** Cerramos el fichero con **fclose()**.

Antes de pasar a los ficheros binarios, probad el ejemplo de escritura en fichero; al pedir las cadenas, introducid en ellas espacios en blanco. ¿Qué ocurre? No se guardan todas las palabras, hay saltos “raros”, ...

Todo esto lo podemos mejorar si usamos la función de lectura de cadenas `gets()`, quedando el programa de la siguiente manera:

```
main.c X
1  #include <stdio.h>
2  #include <string.h>
3
4  int main()
5  {
6      char cadena[50];
7      // Declarar el fichero y abrirlo en modo escritura
8      FILE *f = fopen("fichero.txt", "a");
9      do
10     {
11         printf("Introduce una cadena:");
12         gets(cadena);
13         printf("\n");
14         if(strcmp(cadena, "") != 0)
15         {
16             // Guardar la cadena
17             fputs(cadena, f);
18             // Guardar un salto de línea
19             fputs("\n", f);
20         }
21     } while(strcmp(cadena, "") != 0);
22     printf("Archivo generado correctamente");
23     // Cerrar el fichero
24     fclose(f);
25     return 0;
26 }
```

NOTA:

- Al leer cadenas, mejor usar `gets(cadena)` que `scanf("%s", cadena)`, sobre todo, si hay posibilidad de que el usuario escriba espacios en blanco y no palabras sueltas.
- A partir de ahora, usad siempre `gets()` para leer cadenas.

## Ficheros binarios. Posicionamiento

Los ficheros de texto (plano) pueden ser leídos fácilmente con cualquier editor de texto, por muy básico que sea, pues la estructura de dichos ficheros es estándar para cualquier sistema operativo.

No ocurre lo mismo con los ficheros binarios, que guardan la información con una estructura específica, de modo, que, para leer correctamente dicha información, **debemos conocer perfectamente la estructura con la que se guardó**, si no, es imposible obtener un resultado legible.

Por poner una situación aclaratoria: Si tenemos el siguiente dígito...

10

¿De qué número se trata? ¿Del diez? Eso sería así si trabajamos con la base decimal, pero si estuviéramos trabajando en base binaria, sería el número 2 en decimal ( $1 \cdot 2^1 + 0 \cdot 2^0$ ).

Igual pasa con los ficheros binarios, si no conocemos exactamente la estructura de la información contenida en el mismo, podemos sacarla, pero puede que obtengamos datos erróneos que no tienen nada que ver con la información real.

Las operaciones que podemos hacer sobre ficheros binarios son:

- **fopen()** para abrir el fichero
- **fread()** para sacar datos del fichero
- **fwrite()** para meter datos en el fichero
- **fclose()** para cerrar el fichero

La función que nos permite abrir un fichero es **fopen()**. Sus parámetros son:

- **Nombre** del fichero que queremos abrir, con **path** absoluto o relativo.
- **Modo de apertura** (lo abrimos para leer, para escribir, etc.). Si el modo pone b se abre en binario. Si no lo pone se abre en modo texto. Los posibles modos son:
  - **r** o **rb**: Abre el fichero para lectura. El fichero debe existir o tendremos un error.
  - **w** o **wb**: Abre el fichero para escribir en él. Puede o no existir. Si existe se machaca el contenido que hubiera. Si no existe se crea.
  - **a** o **ab**: Abre el fichero para añadirle datos al final. Respeta el contenido que tuviera.
  - **r+**, **rb+** o **r+b**: Abre el fichero para lectura y escritura (el + es el que indica que es para lectura y escritura a la vez). El fichero debe existir y respeta el contenido, aunque si escribimos algo iremos machacando datos.
  - **w+**, **wb+** o **w+b**: Abre el fichero para lectura y escritura. Crea el fichero si no existe y machaca el contenido si existe.
  - **a+**, **ab+** o **a+b**: Abre el fichero para lectura y escritura. La escritura comenzará al final del fichero, respetando el contenido.

La función **fopen()** nos devuelve un elemento FILE\*. Esto no es más que un **puntero** a una estructura que contiene información sobre el fichero recién abierto. Normalmente podemos ignorar esa información, pero debemos hacer dos cosas con ese FILE\* devuelto:

- **Comprobar que no es NULL.** Si es NULL es que ha habido algún problema y el fichero no se ha podido abrir (se ha abierto un fichero para leer que no existe, se ha abierto para escritura en un directorio en el que no tenemos permiso de escritura, etc.).
- **Guardarlo,** porque es el puntero que requieren las demás funciones para saber sobre qué fichero escribir o leer. Nos servirá, desde el momento que lo abrimos, para identificar el fichero con el que queremos trabajar.

En el siguiente ejemplo, aclaramos cómo trabajar con este tipo de ficheros.



Abriremos un fichero llamado "fichero.dat" al que introduciremos un simple número entero. Cerraremos el fichero y lo volveremos a abrir, ahora para lectura, mostrando finalmente el contenido de este:

```
main.c X
1  #include <stdio.h>
2
3  int main()
4  {
5      // Declarar variables
6      FILE *f1, *f2;
7      int datoEscritura, datoLectura;
8      // Abrir fichero binario modo escritura
9      f1 = fopen ("fichero.dat", "wb");
10     if (f1==NULL)
11     {
12         printf("No se puede abrir fichero.dat");
13         return -1;
14     }
15     else
16     {
17         datoEscritura = 33;
18         fwrite (&datoEscritura, sizeof datoEscritura, 1, f1);
19         printf("Fichero creado correctamente\n");
20         fclose(f1);
21         // Ahora abrimos el fichero, pero con otro puntero
22         // En modo lectura
23         f2 = fopen ("fichero.dat", "rb");
24         if (f2==NULL)
25         {
26             printf("No se puede abrir fichero.dat");
27             return -1;
28         }
29         else
30         {
31             fread (&datoLectura, sizeof datoLectura, 1, f2);
32             printf("Contenido del fichero:%d", datoLectura);
33             fclose(f2);
34             return 0;
35         }
36     }
37 }
```

```
"C:\Users\Studium\Desktop\Jorge\UCAM\Fundame...
Fichero creado correctamente
Contenido del fichero:33
Process returned 0 (0x0)   execution time : 1.899 s
Press any key to continue.
```

## NOTAS:

- **Línea 6:** Declaramos dos punteros a ficheros.
- **Línea 9:** Abrimos el fichero binario en modo escritura.
- **Línea 18:** Metemos un número entero en el fichero. Los parámetros de **fwrite()** son:
  - Dato para guardar
  - Tamaño en bytes del dato para guardar
  - Cantidad
  - Puntero relacionado con el fichero
- **Línea 20:** Cerramos el puntero al fichero.
- **Línea 23:** Abrimos de nuevo el fichero con otro puntero y ahora en modo lectura.
- **Línea 31:** Realizamos una lectura del contenido del fichero. Los parámetros de **fread()** son similares a **fwrite()**.
- **Línea 33:** Cerramos el puntero al fichero.
- Si intentamos abrir el fichero con un editor de texto normal y corriente, la información que se nos muestra no es exactamente la que debería. Esto ocurre porque el editor no sabe cómo debe interpretar los valores binarios y lo hace como buenamente puede, mostrando información errónea.

En el caso de querer guardar cadenas de caracteres podríamos hacer algo similar a esto:

```
...
char *texto;
int nbytes;

texto = "Esto es un texto de prueba";
nbytes = strlen(texto);

fwrite (&nbytes, sizeof nbytes, 1, fh); // Guardamos tamaño de lo siguiente
fwrite (texto, nbytes, 1, fh); // Guardamos el dato, la cadena
..
```

Lo que estamos haciendo es guardar en el fichero primero la longitud de lo que posteriormente vamos a leer.

En todos estos casos, la **lectura** se hace **secuencial**, es decir, se comienza por el primer valor (según el tamaño indicado en el segundo parámetro de **fread()**) y se llega al último pasando por todos los intermedios. O paramos al llegar al elemento buscado. Pero existe la posibilidad de acceder directamente a una posición determinada para leer solamente el dato que allí se encuentra sin pasar y analizar todos los anteriores. Se trata de **fseek()**.

Esta función tiene la siguiente sintaxis:

```
fseek(FILE *puntero, long int salto, int comportamiento);
```

El puntero indica la referencia al fichero físico sobre el que vamos a buscar un dato.

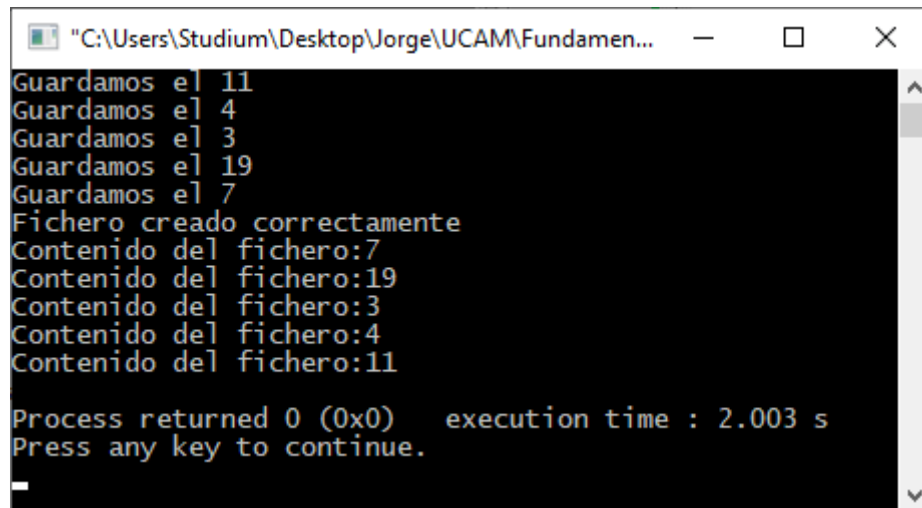
El salto se refiere a la posición en la que queremos colocarnos para sacar el dato. Este salto se calcula multiplicando el tamaño que ocupa cada dato por la cantidad de datos que queramos saltar.

Por último, el comportamiento puede tomar tres posibles valores:

- `SEEK_SET`: El salto se hace desde el principio
- `SEEK_END`: El salto se hace desde el final
- `SEEK_CUR`: El salto se hace desde la posición actual del cursor

Para ver cómo funciona, presentamos un ejemplo que mete 5 enteros en un fichero llamado "datos.dat". Posteriormente, leeremos esos datos en orden inverso gracias a `fseek()`:

```
main.c X
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <time.h>
4
5  int main()
6  {
7      FILE *f1, *f2;
8      int datoEscritura, datoLectura;
9      f1 = fopen ("datos.dat", "wb");
10     if (f1==NULL)
11     {
12         printf("No se puede abrir fichero.dat");
13         return -1;
14     }
15     else
16     {
17         srand(time(NULL)); // Reiniciar semilla aleatoria
18         for(int i = 0; i <5; i++)
19         {
20             datoEscritura = rand()%20+1;
21             printf("Guardamos el %d\n", datoEscritura);
22             fwrite (&datoEscritura, sizeof datoEscritura, 1, f1);
23         }
24         printf("Fichero creado correctamente\n");
25         fclose(f1);
26
27         f2 = fopen ("datos.dat", "rb");
28         if (f2==NULL)
29         {
30             printf("No se puede abrir fichero.dat");
31             return -1;
32         }
33         else
34         {
35             // Colocamos el cursor de lectura al final
36             fseek(f2, -sizeof datoLectura, SEEK_END);
37             for(int i = 0; i <5; i++)
38             {
39                 fread (&datoLectura, sizeof datoLectura, 1, f2);
40                 printf("Contenido del fichero:%d\n", datoLectura);
41                 // Movemos el cursor a su nueva posición
42                 fseek(f2, -2*sizeof datoLectura, SEEK_CUR);
43             }
44             fclose(f2);
45             return 0;
46         }
47     }
48 }
```



```
"C:\Users\Studium\Desktop\Jorge\UCAM\Fundamen...  
Guardamos el 11  
Guardamos el 4  
Guardamos el 3  
Guardamos el 19  
Guardamos el 7  
Fichero creado correctamente  
Contenido del fichero:7  
Contenido del fichero:19  
Contenido del fichero:3  
Contenido del fichero:4  
Contenido del fichero:11  
  
Process returned 0 (0x0)   execution time : 2.003 s  
Press any key to continue.  
_
```

## Referencias

Chuidiang, [enlace](#).

---

14/07/2020