



TEMA 9

Ficheros

Grado en Ingeniería Eléctrica

Grado en Ingeniería Electrónica Industrial y Automática

Escuela Politécnica de Ingeniería de Ferrol

<https://www.udc.es/epef>

9.1.- Introducción

9.2.- Declaración de ficheros

9.3.- Apertura y cierre de ficheros

9.4.- Lectura y escritura de datos

9.5.- Acceso directo a los datos

9.1.- Introducción

- Las estructuras de datos vistas hasta ahora (variables simples, vectores, matrices, etc.) se almacenan en memoria principal
- Por el contrario, los ficheros son estructuras de datos almacenadas en memoria secundaria
- Para utilizar la información en memoria principal se emplea fundamentalmente la instrucción de asignación
- Sin embargo, para guardar o recuperar información de un fichero es necesario realizar una serie de funciones

9.2.- Declaración de ficheros

- El formato de declaración de un fichero es el siguiente:

```
FILE *nombre_variable;
```

- En la declaración no se indica el tipo de datos que se van a almacenar en él.
- Ejemplo:

```
FILE *fichero_datos;
```

9.3.- Apertura y cierre de ficheros

- Antes de usar un fichero es necesario realizar una **operación de apertura** del mismo
- Posteriormente, si se desea almacenar datos en él hay que realizar una **operación de escritura**
- Si se quiere obtener datos de él es necesario hacer una **operación de lectura**
- Cuando ya no se quiera utilizar el fichero se realiza una **operación de cierre** del mismo

9.3.- Apertura y cierre de ficheros

- En la operación de **apertura** se puede decidir si el fichero va a ser de **texto** ó **binario**
- Operación de apertura: **fopen**

```
fichero = fopen(nombre_del_fichero, modo);
```

- El **nombre_del_fichero** será una cadena de caracteres que contenga el nombre (y en su caso la ruta de acceso)
- El **modo** es una cadena de caracteres que indica el tipo del fichero (texto o binario) y el uso que se va a hacer de él (lectura, escritura, añadir datos al final, etc.)

9.3.- Apertura y cierre de ficheros

- Los **modos** disponibles son:
 - **r** : abre un fichero para lectura (si no existe devuelve error)
 - **w** : abre un fichero para escritura (si no existe se crea, si el fichero existe se destruye y se crea uno nuevo)
 - **a** : abre un fichero para añadir datos al final del mismo (si no existe se crea)
 - **+** : símbolo utilizado para abrir el fichero para lectura y escritura
 - **b** : el fichero es de tipo binario
 - **t** : el fichero es de tipo texto. Si no se pone ni b ni t el fichero es de texto
- Los modos anteriores **se pueden combinar** para conseguir abrir el fichero en el modo adecuado

9.3.- Apertura y cierre de ficheros

■ Ejemplos:

- Para abrir un fichero ya existente para lectura y escritura de datos el modo será "r+":

```
fichero = fopen("datos.dat", "r+");
```

- Si el fichero no existe, o aún existiendo se desea crear, el modo será "w+":

```
fichero = fopen("c:\\datos.dat", "w+");
```

- Si deseamos añadir datos al final de un fichero de texto bastará con poner el modo "a":

```
fichero = fopen("c:\\alumnos\\datos.dat", "a");
```


9.3.- Apertura y cierre de ficheros

- Si existe algún **tipo de error** al realizar la operación (por ejemplo, porque se desee abrir para leerlo y éste no exista) devuelve el valor **NULL**
- Por ello, la forma habitual de utilizar la instrucción **fopen** es dentro de una sentencia condicional que permita conocer si se ha producido o no error en la apertura

9.3.- Apertura y cierre de ficheros

- Ejemplo:

```
int main()
{
    FILE *fichero_datos;

    /* Apertura del fichero */
    fichero_datos = fopen("datos.mp3", "rb");

    /* Control del error de apertura */
    if (fichero_datos == NULL)
    {
        printf ("Error: es posible que el fichero no exista");
    }
}
```

9.3.- Apertura y cierre de ficheros

- Cuando se termine el tratamiento del fichero hay que cerrarlo
- Instrucción de cierre: **fclose**

```
fclose(fichero);
```

fichero: variable que apunta al fichero

- Para utilizar las instrucciones de manejo de ficheros es necesario incluir la librería **stdio.h**

9.4.- Lectura y escritura de datos

- Para almacenar datos en un fichero es necesario realizar una operación de escritura
- De igual forma que para obtener datos hay que efectuar una operación de lectura
- En C existen diversas instrucciones para leer y escribir en un fichero; entre ellas: fread-fwrite, fgetc-fputc, fgets-fputs, fscanf-fprintf

9.4.- Lectura y escritura de datos

- Escritura de datos con **fprintf**

- Similar al printf pero hay que especificar como primer argumento el fichero en el que escribimos
- Ejemplos:

```
fprintf (fichero1, "%d", numero);
```

```
fprintf (fichero2, "%c", caracter);
```

```
fprintf (fichero3, "%d %c %s", numero, caracter, cad);
```

9.4.- Lectura y escritura de datos

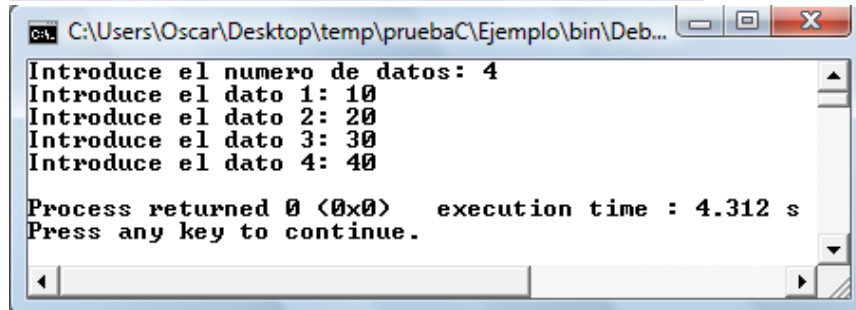
- Ejemplo de escritura una serie de datos enteros en un fichero:

```
#include <stdio.h>

void main()
{
    int numelementos, dato, i;
    FILE *fichero;

    printf("Introduce el numero de datos: ");
    scanf("%i", &numelementos);
    fichero = fopen("datos.txt", "wb");
    if (fichero != NULL)
    {
        for (i=1; i<=numelementos; i++)
        {
            printf("Introduce el dato %i: ", i);
            scanf("%i", &dato);
            fprintf(fichero, "%i ", dato);
        }
        fclose(fichero);
    }
    else
        printf("\nError al abrir el fichero.");
}
```

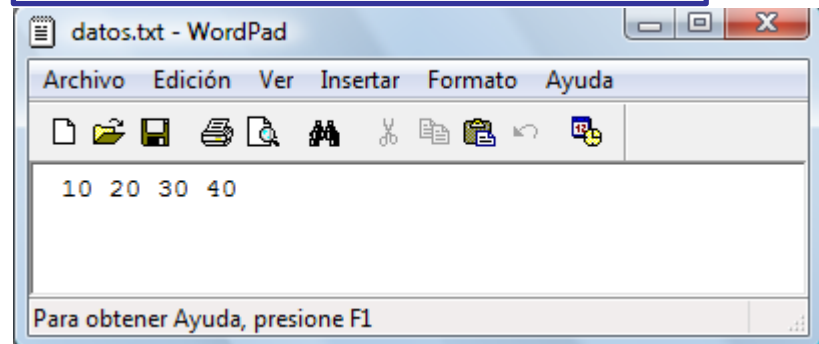
Ejemplo de ejecución:



```
C:\Users\Oscar\Desktop\temp\pruebaC\Ejemplo\bin\Deb...
Introduce el numero de datos: 4
Introduce el dato 1: 10
Introduce el dato 2: 20
Introduce el dato 3: 30
Introduce el dato 4: 40

Process returned 0 (0x0)   execution time : 4.312 s
Press any key to continue.
```

Datos almacenados en el fichero:



```
datos.txt - WordPad
Archivo  Edición  Ver  Insertar  Formato  Ayuda

10 20 30 40

Para obtener Ayuda, presione F1
```

9.4.- Lectura y escritura de datos

■ Lectura de datos con **fscanf**

- Similar al `scanf` pero hay que especificar como primer argumento el fichero en el que leemos

- Ejemplos:

```
fscanf (fichero1, "%d", &numero);
```

```
fscanf (fichero2, "%c", &character);
```

```
fscanf (fichero3, "%d %c %s", &numero, &character, cad);
```

- Para leer todos los datos de fichero: lecturas sucesivas hasta que se lee el **final del fichero**
- Función `feof(fichero)` → devuelve valor `!= 0` si se alcanza el final de fichero

9.4.- Lectura y escritura de datos

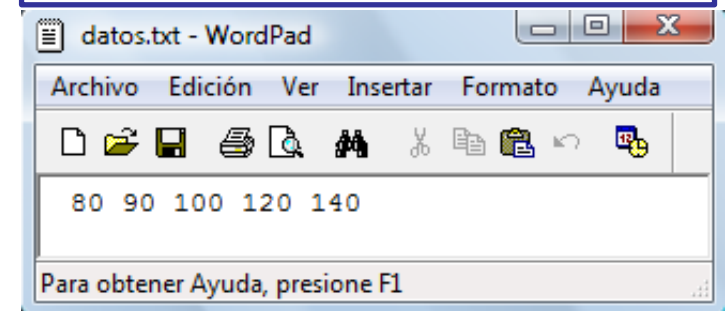
- Ejemplo de la lectura de los datos (enteros) almacenados en un fichero:

```
#include <stdio.h>

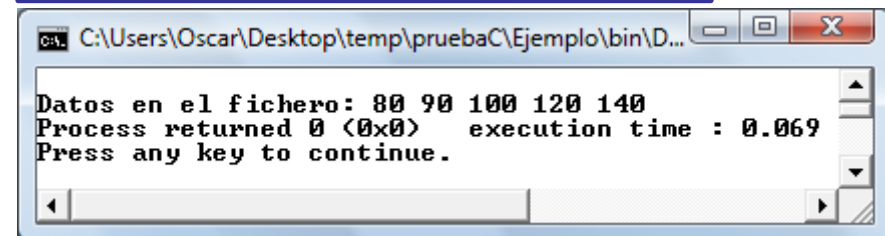
void main()
{
    int dato;
    FILE *fichero;

    fichero = fopen("datos.txt", "rb");
    if (fichero != NULL)
    {
        printf("\nDatos en el fichero: ");
        while ( !feof(fichero) )
        {
            fscanf(fichero, "%i", &dato);
            printf("%i ", dato);
        }
        fclose(fichero);
    }
    else
        printf("\nError al abrir el fichero");
}
```

Datos almacenados en el fichero:



Ejemplo de ejecución:



9.4.- Lectura y escritura de datos

- Ejemplo de almacenamiento de los datos de un fichero en un vector:

```
#include <stdio.h>
#define MAX 50
void main()
{
    FILE *fichero;
    int vector[MAX], posicion=0;

    fichero = fopen("datos.txt", "rb");
    if (fichero != NULL)
    {
        while(!feof(fichero) && posicion < MAX) <-----
        {
            fscanf(fichero, "%d\n", &vector[posicion]);
            printf("El dato en la posicion %i es: %i\n", posicion, vector[posicion]);
            posicion = posicion + 1;
        }
        fclose(fichero);
    }
    else
        printf("\nError al abrir el fichero\n");
}
```

El bucle se repite mientras no se alcance el final de fichero y el vector no esté lleno (si la posición actual es menor que la del máximo número de elementos)

9.4.- Lectura y escritura de datos

- Ejemplo con dos ficheros: ¿qué realiza este programa?

```
#include <stdio.h>
#include <string.h>
#define MAX 80
void main()
{
    char nombre[MAX], nombreFichero[MAX];
    FILE *fichero1, *fichero2;

    printf("\nIntroduce el nombre del fichero 1: ");
    scanf("%79s", nombreFichero); <-----
    fichero1 = fopen(nombreFichero, "r");
    if (fichero1 != NULL)
    {
        printf("\nIntroduce el nombre del fichero 2: ");
        scanf("%79s", nombreFichero);
        fichero2 = fopen(nombreFichero, "w");
```

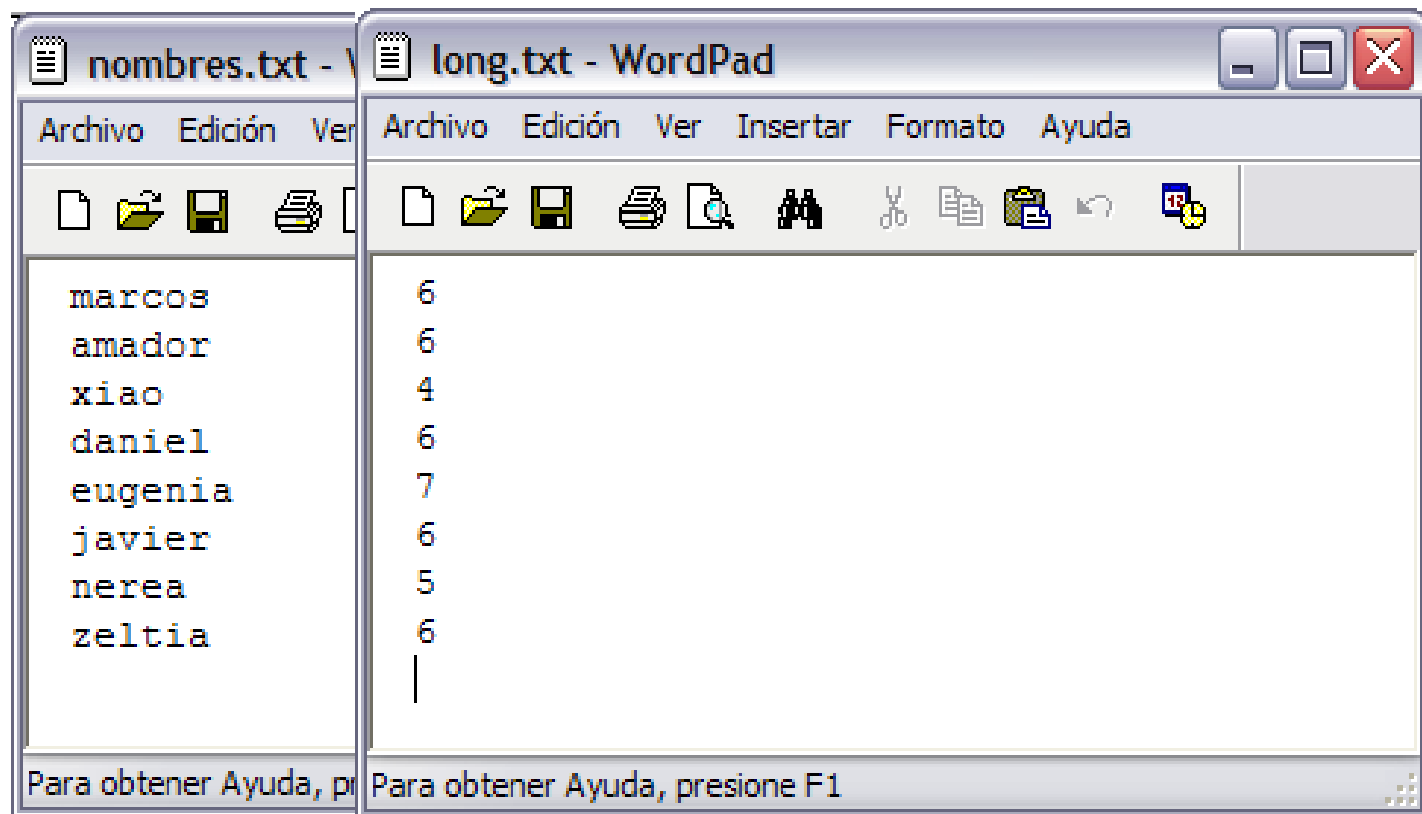
(continúa en la siguiente columna)

Se lee una cadena de caracteres que indicará el nombre del primer fichero

```
        if (fichero2 != NULL)
        {
            while (!feof(fichero1))
            {
                fscanf(fichero1, "%s", nombre);
                fprintf(fichero2, "%i\n", strlen(nombre));
            }
            fclose(fichero2);
        }
        else
            printf("\nError al crear el fichero\n");
            fclose(fichero1);
    }
    else
        printf("\nError al abrir el fichero\n");
}
```

9.4.- Lectura y escritura de datos

- Ejemplo del resultado proporcionado por el programa anterior para un fichero dado llamado *nombres.txt*:



9.5.- Acceso directo a los datos

- Para desplazarse en el fichero hasta una determinada posición deseada:

fseek (fichero, posicion, origen);

Coloca el puntero del *fichero* a tantos bytes del *origen* como indica *posición*

Los posibles orígenes son:

SEEK_SET ó 0 : principio del fichero

SEEK_CUR ó 1 : posición actual

SEEK_END ó 2 : final del fichero

- Ejemplo en un fichero que almacene números enteros:

fseek (fichero, 3*sizeof(int), SEEK_CUR);

Colocará el puntero 3 posiciones más allá de la posición actual del puntero

En este ejemplo se ha usado la función **sizeof(tipo_dato)** que devuelve un valor entero que indica el tamaño que ocupa el tipo de dato especificado

Ejemplo adicional con uso de funciones (parte I)

- Programa que lee unos datos enteros de un fichero y almacena en otro fichero aquellos datos, del primer fichero, que no son negativos

```
#include <stdio.h>
#define MAX 61
```

```
FILE * abrir_fichero(char modo[]) <----- Función que abre un fichero con cualquier modo (lectura,
{
    FILE *fichero;
    char nombre_fichero[MAX];
    do
    {
        printf("Introduce el nombre de fichero: ");
        scanf("%60s", nombre_fichero); <----- Se lee por teclado una cadena de caracteres que
                                                indicará el nombre del fichero a abrir. Esta
                                                cadena se usa en la siguiente instrucción (fopen)
                                                para indicar el nombre de fichero.

        fichero = fopen(nombre_fichero, modo);
        if (fichero == NULL)
            printf("\nError al abrir el fichero. Intentalo de nuevo.");
    } while(fichero == NULL); <----- Si no se ha abierto correctamente el fichero se
                                                vuelve a repetir todo el proceso.

    return fichero; <----- La función devuelve como resultado la dirección de memoria
}                                     que apunta al fichero (el puntero al fichero).
```

Ejemplo adicional con uso de funciones (parte II)

```
void filtrar_negativos(FILE *fichero1, FILE *fichero2)
{
    int dato;

    while(!feof(fichero1))
    {
        fscanf(fichero1, "%i", &dato); <----- En el primer fichero que se le pasa a la función se
        if (dato >= 0)                                     leen datos enteros.
            fprintf(fichero2, "%i\n", dato); <----- En el segundo fichero que se le pasa a la función se
    }                                                         escriben los datos que no sean negativos.
}

int main()
{
    FILE *fich1, *fich2;

    fich1 = abrir_fichero("r"); <----- Abre el primer fichero para leer datos de él
    fich2 = abrir_fichero("w"); <----- (por ello se le pasa una cadena "r").
                                     Abre el segundo fichero para escribir datos en él
                                     (por ello se le pasa una cadena "w").

    filtrar_negativos(fich1, fich2); <----- Se llama a la función que lee datos del primer fichero y
                                     almacena en el segundo sólo aquellos datos que no son
                                     negativos (para ello se le pasan como argumentos los
                                     punteros a ambos ficheros).

    fclose(fich1); <----- Se cierran ambos ficheros.
    fclose(fich2);
    return 0;
}
```