❑ FICHEROS

✓ Para el lenguaje C un flujo de datos es una corriente o flujo de bytes que puede hacerse corresponder con un fichero físico en disco.

✓ Para utilizar ficheros en primer lugar se debe declarar un flujo de datos como FILE * siendo el tipo FILE una estructura definida en la cabecera <stdio.h> (así como el resto de funciones para manejar ficheros).

✓ Los ficheros pueden ser:

- Ficheros de Texto: si están divididos en líneas que contienen caracteres y que acaban en un salto de línea.
- Ficheros binarios: Si tienen cualquier otra estructura.

✓ Antes de trabajar con un fichero debemos abrirlo. Esta operación conecta el flujo con el fichero físico en disco. A partir de ese momento trabajaremos con el fichero a través del flujo con el que está conectado. En la operación de apertura es dónde se decide de qué tipo será el fichero (binario o texto).

- ✓ Una vez que se acabe de trabajar con un fichero este debe cerrarse, desconectando así el flujo del fichero físico.

✓ Declaración de un flujo

FILE * id_var_fichero;

✓ Apertura y cierre de un fichero

Antes de poder leer o escribir datos en un fichero hay que abrirlo mediante la función fopen() definida del siguiente modo:

FILE * fopen (const char *nombre_fichero, const char *modo_apertura)

Como parámetros recibe dos cadenas:

Nombre_fichero: es el nombre o ruta de acceso del fichero físico que deseamos abrir.

Modo_apertura: especifica el modo de apertura, indica el tipo de fichero (texto o binario) y el uso que se va a hacer de él lectura, escritura, añadir datos al final, etc.

Los modos disponibles son:

Modo	Descripción
r	abre un fichero para lectura. Si el fichero no existe devuelve error.
w	abre un fichero para escritura. Si el fichero 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 combinan para conseguir abrir el fichero en el modo adecuado.

Por ejemplo, para abrir un fichero binario ya existente para lectura y escritura el modo será "rb+ "; si el fichero no existe, o aun existiendo se desea crear, el modo será " wb+". Si deseamos añadir datos al final de un fichero de texto bastará con poner "a", etc.

La función fopen() devuelve un flujo que usaremos para trabajar con el fichero, a través del resto de funciones de manejo de ficheros, o NULL en caso de que el fichero no se haya podido abrir.

Ejemplo

Abre un fichero de texto existente para lectura

```
#include <stdio.h>
#include <stdlib.h>
…
FILE *pf
…
if ((pf= fopen("prueba.txt", "r"))==NULL)
{
printf("Error de apertura\n");
exit(1);
}
```

Una vez que se ha terminado de utilizar el fichero debemos cerrarlo con la siguiente función:

int fclose(FILE* id_var_fichero);

Devuelve 0 si el cierre se ha realizado con éxito y la macro EOF en caso contrario.

✓ Funciones de lectura

- Lectura de caracteres de un fichero:

int fgetc (FILE *f);

Devuelve el carácter del fichero situado en la posición actual o EOF si es final del fichero. Avanza el indicador a la siguiente posición.

- Lectura de cadenas de un fichero:

char * fgets(char * s, int tam, FILE * f);

Lee una cadena de caracteres del fichero y la copia en la cadena apuntada por s, el tamaño de la cadena leida vendrá determinado por el entero tam, por el carácter fin de línea o por el carácter de fin de fichero EOF. Devuelve un puntero a la misma cadena.

- Lectura formateada de un fichero:

int fscanf(FILE * f, const char * formato, …);

Funciona exactamente igual que scanf pero leyendo del fichero en vez de de la entrada estándar (flujo stdin).

- Lectura de bloques de un fichero:

size_t fread (void * p, size_t tam, size_t n, FILE * f);

Lee tantos datos como indique **n** del fichero, colocando los datos leídos a partir de la dirección **p**. Los datos tienen que tener tantos bytes como especifique **tam**. La función fread devuelve el número de elementos leídos, y el valor devuelto debe coincidir con **n**.

✓ Funciones de escritura

- Escritura de caracteres en un fichero:

int fputc (int c, FILE *f);

Escribe el carácter c en la posición actual del fichero. Devuelve el carácter escrito o EOF en caso de error. Avanza el indicador a la siguiente posición.

- Escritura de cadenas en un fichero:

int fputs(const char * s, FILE * f);

Escribe la cadena de caracteres apuntada por s en el fichero. Devuelve EOF si se produce un error.

- Escritura formateada en un fichero:

int fprintf(FILE * f, const char * formato, …);

Funciona exactamente igual que printf pero escribiendo en el fichero en vez de de la salida estándar (flujo stdout).

- Escritura de bloques en un fichero:

size_t fwrite (const void * p, size_t tam, size_t n, FILE * f);

Escribe tantos datos como indique **n** el fichero, tomando los datos a partir de **p**. Cada dato leido tiene que tener tantos bytes como especifique **tam**. La función **fwrite** devuelve el número de elementos escritos, este valor debe coincidir con **n**.

Las funciones de lectura y escritura de bloques suelen usarse con ficheros binarios, para permitir leer o escribir datos de cualquier tipo: números, caracteres, o incluso estructuras completas: es decir que nos permitan leer o escribir zonas de memoria que puedan contener cualquier tipo de datos. Son funciones muy potentes, con una sola llamada podemos leer o escribir un vector, una estructura o incluso un vector de estructuras.

Ejemplo 1:

```
FlLE *f;
int v[6], elem_escritos, num;
f = fopen (“datos.cht ", "wb ");
/* Para escribir los 3 últimos elementos de v (el 2, el 3 y el 4) */
elem-escritos = fwrite (&v[2], sizeof(int), 3, f );
/* Para escribir el primer elemento de v, valen las 2 instrucciones
siguientes */
fwrite (v, sizeof (int), 1, f );
fwrite (&v[0], sizeof(int), 1, f );
/* Para escribir un entero valen las dos siguientes */
fwrite (&num, sizeof(int), 1, f);
fwrite (&num, sizeof(num), 1, f);
```

Ejemplo 2:

```
f = fopen (“datos.dat ", "rb ");
elem-escritos = fread (&v[2], sizeof(int), 3, f);
fread (v, sizeof(int), 1, f);
fread (&V[0], sizeof(int), 1, f);
fread (&num, sizeof(int), 1, f);
fread (&num, sizeof(num), 1, f);
```

✓ Funciones de posicionamiento

- Conocer la posición del indicador:

long int ftell (FILE *f);

Para ficheros binarios devuelve la posición del indicador en número de bytes medida desde el principio.

- Posicionamiento directo:

int fseek (FILE * f, long int adonde, int desdedonde);

Posiciona el indicador en la posición **adonde**, medida en bytes desde donde indique el argumento **desdedonde**. Teniendo en cuenta las siguientes constantes de desplazamiento:

SEEK_SET desde el principio
SEEK_CUR desde donde estamos
SEEL_END desde el final

- Rebobinamiento:

void rewind (FILE *f);

Posiciona el indicador al principio del fichero.