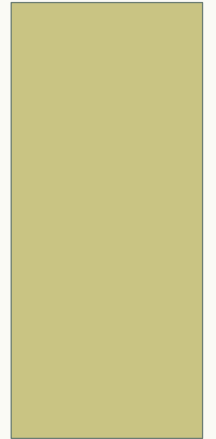


ARCHIVOS DE TEXTO

ACCESO SECUENCIAL



MANEJO DE ENTRADA/SALIDA

- Toda la entrada/salida se maneja por medio de **flujos o streams**, los cuales son secuencias de bytes.
- En **operaciones de entrada** los bytes fluyen desde un dispositivo (por ej: el teclado, el disco duro, una conexión de red, etc.) hacia la memoria principal.
- En **operaciones de salida** los bytes fluyen desde la memoria principal hacia un dispositivo (por ej: la pantalla, una impresora, un disco, una conexión de red, etc.)

MANEJO DE ENTRADA/SALIDA

- Cuando inicia el programa se conectan tres flujos automáticamente:
 - El flujo estándar de entrada al teclado.
 - El flujo estándar de salida a la pantalla.
 - El flujo estándar de error a la pantalla.
- Estos flujos se pueden redireccionar.

ARCHIVOS

- C ve a un archivo como una secuencia de bytes. Cuando se abre el archivo se le asocia un flujo o stream.
- Desde un programa C se puede (stdio.h)
 - Crear archivos
 - Leer la información que contiene
 - Modificar el contenido del archivo.
 - Eliminar y renombrar archivos

ARCHIVOS Y DISPOSITIVOS ESTANDAR

- La biblioteca **stdio.h** ofrece funciones para manipulación de E/S
 - Usando los dispositivos estándar
 - Usando archivos
- Para evitar diferencia en las operaciones, la biblioteca stdio.h trata a todos como archivos incluyendo los dispositivos de E/S estándar: Teclado y monitor

CÓMO OPERAR CON ARCHIVOS?

- Para trabajar con archivos se debe
 - Declarar una variable de tipo **FILE *** (un puntero a FILE)
 - La variable de tipo **FILE** sirve para representar a un archivo en el programa C.
 - Debe declararse una variable FILE * por cada archivo a utilizar.
 - Asociar dicha variable al archivo utilizando la función **fopen** (abre el archivo)
 - Operar sobre el archivo.
 - Al finalizar, cerrar el archivo utilizando **fclose**.

DISPOSITIVOS ESTANDAR

- Existen tres identificadores especiales de tipo **FILE ***
 - **stdin**: dispositivo estándar de entrada (teclado)
 - **stdout** : dispositivo estándar de salida (el monitor)
 - **stderr**: dispositivo estándar de salida de errores (el monitor)

FUNCIÓN `fopen`

`FILE * fopen(const char *nombre, const char *modo);`

- Abre un archivo cuyo nombre es la cadena apuntada por **`nombre`**, y asigna un flujo a ello.
- El argumento **`modo`** apunta a una cadena de caracteres que indican el modo de apertura.
- Si la apertura del archivo falla retorna NULL.

EJEMPLO

```
FILE * arch;  
arch = fopen("prueba.txt", "w");
```


FUNCION `fclose`

`int fclose(FILE * miArchivo);`

- El identificador pasado como parámetro será desasociado del archivo.
- La función *fclose* retorna cero si el archivo fue cerrado con éxito. Si se detectaron errores, entonces retorna EOF.

Para trabajar con el archivo lo primero que hay que hacer es abrirlo

```
#include <stdio.h>
int main()
{   FILE *archivo;
    char nombre[10] = "datos.dat";

    archivo = fopen( nombre, "w" );
    printf( "Archivo: %s -> ", nombre );
    if( archivo )
        printf( "creado (ABIERTO)\n" );
    else {
        printf( "Error (NO ABIERTO)\n" );
        return 1;
    }

    if( !fclose(archivo) )
        printf( "Archivo cerrado\n" );
    else {
        printf( "Error: fichero NO CERRADO\n" );
        return 1;
    }
    return 0;
}
```

01_fopen_fclose_v1.c

```
#include <stdio.h>
```

```
int main()
```

```
{ FILE *archivo;
```

```
char nombre[10] = "datos.dat";
```

Modo de apertura
del archivo

```
archivo = fopen( nombre, "w" );
```

```
printf( "Archivo: %s -> ", nombre );
```

```
if( archivo )
```

```
    printf( "creado (ABIERTO)\n" );
```

```
else {
```

```
    printf( "Error (NO ABIERTO)\n" );
```

```
    return 1;
```

```
}
```

```
if( !fclose(archivo) )
```

```
    printf( "Archivo cerrado\n" );
```

```
else {
```

```
    printf( "Error: fichero NO CERRADO\n" );
```

```
    return 1;
```

```
}
```

```
return 0;
```

```
}
```

01_fopen_fclose_v1.c

```
#include <stdio.h>
int main()
{   FILE *archivo;
    char nombre[10] = "datos.dat";

    archivo = fopen( nombre, "w" );
    printf( "Archivo: %s -> ", nombre );
    if( archivo )
        printf( "creado (ABIERTO)\n" );
    else {
        printf( "Error (NO ABIERTO)\n" );
        return 1;
    }

    if( !fclose(archivo) )
        printf( "Archivo cerrado\n" );
    else {
        printf( "Error: fichero NO CERRADO\n" );
        return 1;
    }
    return 0;
}
```

archivo valdrá NULL (cero) si hubo error y un valor distinto de cero si no

```

#include <stdio.h>
int main()
{
    FILE *archivo;
    char nombre[10] = "datos.dat";

    archivo = fopen( nombre, "w" );
    printf( "Archivo: %s -> ", nombre );
    if( archivo )
        printf( "creado (ABIERTO)\n" );
    else {
        printf( "Error (NO ABIERTO)\n" );
        return 1;
    }

    if( !fclose(archivo) ) ←
        printf( "Archivo cerrado\n" );
    else {
        printf( "Error: fichero NO CERRADO\n" );
        return 1;
    }
    return 0;
}

```

fclose cierra el archivo.
Retorna cero si no
hubo problemas o
EOF (-1) si falló

```

#include <stdio.h>
int main()
{
    FILE *archivo;
    char nombre[10] = "datos.dat";

    archivo = fopen( nombre, "w" );
    printf( "Archivo: %s -> ", nombre );

    if( archivo == NULL) {
        printf( "Error (NO ABIERTO)\n" );
        return 1;
    }
    printf( "creado (ABIERTO)\n" );

    if (fclose(archivo)) {
        printf( "Error: archivo NO CERRADO\n" );
        return 1;
    }
    printf( "Archivo cerrado\n" );

    return 0;
}

```

Puede
reemplazarse por
! archivo ?

FUNCIÓN `fprintf`

`int fprintf(FILE * arch, const char *formato, ...);`

- Envía datos al stream apuntado por **`arch`**, bajo el control de la cadena apuntada por **`formato`** que especifica cómo los argumentos posteriores son convertidos para la salida.
- Si hay argumentos insuficientes para el formato, el comportamiento no está definido.
- Si el formato termina mientras quedan argumentos, los argumentos restantes son evaluados (como siempre) pero ignorados.

FUNCIÓN `fprintf`

`int fprintf(FILE * arch, const char *formato, ...);`

- La función retorna el control cuando el final de la cadena de formato es encontrado.
- Retorna el número de caracteres transmitidos, o un valor negativo si un error de salida se ha producido.

Ejemplos

- `fprintf(miArchivo, "Usando la funcion \'fprintf\'\\n");`
- `fprintf(miArchivo, "%d \t %d \t %d", n1, n2, n3);`


```
#include <stdio.h>
```

```
int main()
```

```
{ FILE * arch;
```

```
arch = fopen("prueba.txt", "w");
```

Abre el
archivo

```
if (arch == NULL) {
```

```
    fprintf(stdout, "Error al abrir el archivo!\n");
```

```
    return 1;
```

```
}
```

```
fprintf(stdout, "El archivo está abierto\n");
```

```
fprintf(arch, "Este es mi primer archivo \n");
```

```
fprintf(arch, "creado desde un programa C.");
```

```
fclose(arch);
```

```
fprintf(stdout, "El archivo está cerrado\n");
```

```
return 0;
```

02_fprintf.c

```
}
```

```
#include <stdio.h>
```

```
int main()
```

```
{ FILE * arch;
```

Modo de apertura del
archivo



```
arch = fopen("prueba.txt", "w");
```

```
if (arch == NULL) {
```

```
    fprintf(stdout, "Error al abrir el archivo!\n");
```

```
    return 1;
```

```
}
```

```
fprintf(stdout, "El archivo está abierto\n");
```

```
fprintf(arch, "Este es mi primer archivo \n");
```

```
fprintf(arch, "creado desde un programa C.");
```

```
fclose(arch);
```

```
fprintf(stdout, "El archivo está cerrado\n");
```

```
return 0;
```

```
}
```

02_fprintf.c

```
#include <stdio.h>
```

```
int main()
```

```
{ FILE * arch;
```

stdout es la salida estándar, por lo tanto, esto sale por pantalla.

```
arch = fopen("prueba.txt", "w");
```

```
if (arch == NULL) {  
    fprintf(stdout, "Error al abrir el archivo!\n");  
    return 1;  
}
```

```
fprintf(stdout, "El archivo está abierto\n");
```

```
fprintf(arch, "Este es mi primer archivo \n");
```

```
fprintf(arch, "creado desde un programa C.");
```

```
fclose(arch);
```

```
fprintf(stdout, "El archivo está cerrado\n");
```

```
return 0;
```

02_fprintf.c

```
}
```

```
#include <stdio.h>
```

```
int main()
```

```
{ FILE * arch;
```

```
arch = fopen("prueba.txt", "w");
```

```
if (arch == NULL) {
```

```
    fprintf(stdout, "Error al abrir el archivo!\n");
```

```
    return 1;
```

```
}
```

```
fprintf(stdout, "El archivo está abierto\n");
```

```
fprintf(arch, "Este es mi primer archivo \n");
```

```
fprintf(arch, "creado desde un programa C.");
```

```
fclose(arch);
```

```
fprintf(stdout, "El archivo está cerrado\n");
```

```
return 0;
```

```
}
```

Esto se guarda en el archivo

02_fprintf.c

```
#include <stdio.h>
#include <stdio.h>
```

```
int main()
```

```
{ FILE * arch;
```

```
arch = fopen("prueba.txt", "w");
```

```
if (arch==NULL)
```

```
    fprintf(stdout, "Error al abrir el archivo!\n");
```

```
else {
```

```
    fprintf(stdout, "El archivo está abierto\n");
```

```
    fprintf(arch, "Este es mi primer archivo \n");
```

```
    fprintf(arch, "creado desde un programa C.");
```

```
    fclose(arch);
```



```
    fprintf(stdout, "El archivo está cerrado\n");
```

```
}
```

```
return 0;
```

```
}
}
```

Cuando se termina de utilizar el archivo hay que cerrarlo.

02_fprintf.c

```
#include <stdio.h>
```

```
int main()
```

```
{ FILE *arch;
```

```
char nombre[10] = "datos.txt";
```

```
unsigned int i;
```

```
arch = fopen( nombre, "w" );
```

```
if( !arch ) {
```

```
    printf( "Error (NO ABIERTO)\n" );
```

```
    return 1;
```

```
}
```

```
fprintf( arch, "Ejemplo de la funcion \'fprintf\'\n" );
```

```
fprintf( arch, "\t 2\t 3\t 4\n" );
```

```
fprintf( arch, "x\tx\tx\tx\n\n" );
```

```
for( i=1; i<=10; i++ )
```

```
    fprintf( arch, "%d\t%d\t%d\t%d\n", i, i*i, i*i*i,  
            i*i*i*i );
```

```
fprintf( stdout, "Datos guardados en el archivo: %s\n",  
        nombre );
```

```
fclose(arch);
```

```
return 0;
```

```
}
```

¿Qué muestra en
pantalla?

```
#include <stdio.h>
```

```
int main()
```

```
{ FILE *arch;
```

```
  char nombre[10] = "datos.txt";
```

```
  unsigned int i;
```

```
  arch = fopen( nombre, "w" );
```

```
  if( !arch ) {
```

```
    printf( "Error (NO ABIERTO)\n" );
```

```
    return 1;
```

```
  }
```

```
  fprintf( arch, "Ejemplo de la funcion \'fprintf\'\n" );
```

```
  fprintf( arch, "\t 2\t 3\t 4\n" );
```

```
  fprintf( arch, "x\tx\tx\tx\n\n" );
```

```
  for( i=1; i<=10; i++ )
```

```
    fprintf( arch, "%d\t%d\t%d\t%d\n", i, i*i, i*i*i,  
              i*i*i*i );
```

```
  fprintf( stdout, "Datos guardados en el archivo: %s\n",  
           nombre );
```

```
  fclose(arch);
```

```
  return 0;
```

```
}
```

¿Qué guarda en el
archivo *datos.txt*?

ARCHIVO datos.txt

Ejemplo de la funcion 'fprintf'

	2	3	4
x	x	x	x
1	1	1	1
2	4	8	16
3	9	27	81
4	16	64	256
5	25	125	625
6	36	216	1296
7	49	343	2401
8	64	512	4096
9	81	729	6561
10	100	1000	10000

MODOS DE APERTURA DE ARCHIVO

Modo	Descripción
r	Abrir un archivo para lectura.
w	Crear un archivo para escritura. Si el archivo ya existe, se descarta el contenido actual.
a	Abrir o crear un archivo para escribir al final del mismo
r+	Abrir un archivo para lectura y escritura.
w+	Genera un archivo para lectura y escritura. Si el archivo ya existe, se descarta el contenido actual.
a+	Abrir o crear un archivo para actualizar. La escritura se efectuará al final del archivo.

EJERCICIO 1

- Escriba un programa en C que permita agregar líneas de texto al final del archivo "prueba.txt".
El programa termina al ingresar la palabra "FIN".

FUNCIÓN `fscanf`

`int fscanf(FILE *arch, const char *formato, ...);`

- Recibe datos del stream apuntado por **`arch`**, bajo el control de la cadena apuntada por **`formato`** que especifica las secuencias de entrada permitidas y cómo han de ser convertidas para la asignación.
- Si hay argumentos insuficientes para el formato, el comportamiento no está definido.
- Si el formato termina mientras quedan argumentos, los argumentos restantes son evaluados (como siempre) pero ignorados.

FUNCIÓN *fscanf*

int fscanf(FILE *arch, const char *formato, ...);

- La función retorna control cuando el final de la cadena de formato es encontrado.
- Retorna el número de datos de entrada asignados, que puede ser menor que el ofrecido, incluso cero, en el caso de un error de asignación.
- Si un error de entrada ocurre antes de cualquier conversión, la función *fscanf* retorna EOF.

```
#include <stdio.h>
```

```
int main()
```

```
{ FILE *arch;  
  char nombre[10] = "datos.txt";  
  unsigned int i, x1, x2, x3, x4;
```

Muestra en pantalla el
contenido del archivo
datos.txt

```
arch = fopen( nombre, "r" );  
printf( "Datos leidos del archivo: %s\n", nombre );  
printf( "Ejemplo de la funcion \'fprintf\'\n" );  
printf( "\t 2\t 3\t 4\n" );  
printf( "x\tx\tx\tx\n\n" );
```

```
fscanf( arch, "Ejemplo de la funcion \'fprintf\'\n" );  
fscanf( arch, "\t 2\t 3\t 4\n" );  
fscanf( arch, "x\tx\tx\tx\n\n" );  
for( i=1; i<=10; i++ )  
{ fscanf( arch, "%d\t%d\t%d\t%d\n", &x1, &x2, &x3, &x4 );  
  printf( "%d\t%d\t%d\t%d\n", x1, x2, x3, x4 );  
}  
fclose(arch);  
return 0;
```

```
}
```

```
#include <stdio.h>
```

```
int main()
```

```
{ FILE *arch;  
  char nombre[10] = "datos.txt";  
  unsigned int i, x1, x2, x3, x4;
```

fscanf es equivalente a scanf pero toma la información del archivo

```
  arch = fopen( nombre, "r" );  
  printf( "Datos leídos del archivo: %s\n", nombre );  
  printf( "Ejemplo de la función 'fprintf'\n" );  
  printf( "\t 2\t 3\t 4\n" );  
  printf( "x\tx\tx\tx\n\n" );
```

```
  fscanf( arch, "Ejemplo de la función 'fprintf'\n" );  
  fscanf( arch, "\t 2\t 3\t 4\n" );  
  fscanf( arch, "x\tx\tx\tx\n\n" );
```

```
  for( i=1; i<=10; i++ )  
  { fscanf( arch, "%d\t%d\t%d\t%d\n", &x1, &x2, &x3, &x4 );  
    printf( "%d\t%d\t%d\t%d\n", x1, x2, x3, x4 );  
  }  
  fclose( arch );  
  return 0;
```

```
}
```

FUNCIÓN **feof**

- En el caso que estemos leyendo un archivo, será útil saber si este ya terminó.
- La función **feof** indica si la última operación realizada sobre el archivo excedió el final de este.

- **Sintaxis**

`int feof(FILE *stream);`

- La función retorna un número diferente a 0 (TRUE) si el archivo terminó y 0 (FALSE) en otro caso.

```
#include <stdio.h>
```

```
int main()
```

```
{ FILE * Ptr;
```

```
float fn1, fn4;
```

```
int n2, n3;
```

fscanf toma la información
del archivo

```
Ptr = fopen ("numeros.txt", "r");
```

```
if ( Ptr ) {
```

```
printf("El archivo \"Numeros.txt\" esta abierto\n") ;
```

```
fscanf(Ptr, "%f %d %d %f", &fn1, &n2, &n3, &fn4);
```



```
while (! feof(Ptr))
```

```
{ printf("%.1f %d %d %.1f\n", fn1, n2, n3, fn4);
```

```
fscanf(Ptr, "%f %d %d %f", &fn1, &n2, &n3, &fn4);
```

```
}
```

```
fclose (Ptr);
```

```
}
```

```
else printf("Error al abrir \"Numeros.txt\"\\n");
```

```
return 0;
```

```
}
```

fscanf.c


```
#include <stdio.h>
```

```
int main()
```

```
{ FILE * Ptr;
```

```
float fn1, fn4;
```

```
int n2, n3;
```

```
Ptr = fopen ("numeros.txt", "r");
```

```
if ( Ptr ) {
```

```
printf("El archivo \"Numeros.txt\" esta abierto\n") ;
```

```
fscanf(Ptr, "%f %d %d %f", &fn1, &n2, &n3, &fn4);
```

```
while (! feof(Ptr))
```



```
{ printf("%.1f %d %d %.1f\n", fn1, n2, n3, fn4);
```

```
fscanf(Ptr, "%f %d %d %f", &fn1, &n2, &n3, &fn4);
```

```
}
```

```
fclose (Ptr);
```

```
}
```

```
else printf("Error al abrir \"Numeros.txt\"\\n");
```

```
return 0;
```

feof permite saber si nos
pasamos del límite del
archivo

fscanf.c

FUNCIÓN **feof**

- Note que la función **feof** indica si ya se realizó una operación fuera del límite del archivo; no si se encuentra posicionado en el límite del archivo.
- Por este motivo, en el ejemplo anterior, para leer los caracteres del archivo utilizamos:

```
/* operación de lectura */  
while (! feof( arch )){  
    /* procesamiento de los datos */  
    /* operación de lectura */  
}
```

FUNCION **fgetc**

- **Sintaxis**

int fgetc(FILE *stream);

- Lee un carácter (si existe) desde el stream de entrada apuntado por **stream**.
- El valor retornado es un **unsigned char** convertido a **int**. Si hubo un error, retorna EOF
- El indicador de posición de ficheros asociado al stream es incrementado en una posición (si está definido).
- **fgetc(stdin)** equivale a **getchar()**

FUNCION **fputc**

- **Sintaxis**

int fputc(int c, FILE *stream);

- Escribe el carácter indicado por **c** (convertido a un **unsigned char**) al stream de salida apuntado por **stream**.
- Escribe en la posición indicada por el indicador de posición de ficheros asociado al stream (si está definido), y avanza el indicador apropiadamente.
- Retorna el carácter escrito. Si ocurre un error de escritura, retorna EOF.
- **fputc('a', stdout)** equivale a **putchar('a')**

EJERCICIO

- Escriba un programa C que muestre en pantalla el contenido de un archivo de texto.
- Utilice la función **fgetc** para leer cada carácter y la función **feof** para reconocer el fin de archivo.

EJERCICIO

- Escriba un programa C que compare el contenido de dos archivos de texto denominados "Lectura1.txt" y "Lectura2.txt".
- Si no son iguales, el programa deberá imprimir la ubicación del primer carácter diferente (número de línea y número de carácter dentro de la línea).

FUNCION **fgets**

- **Sintaxis**

`char *fgets(char *cadena, int n, FILE *stream);`

- Lee como máximo (**n-1**) caracteres desde el stream apuntado por **stream** al array apuntado por **cadena**.
- Ningún carácter adicional es leído después del carácter de nueva línea (el cual es retenido) o después de un final de fichero (EOF).
- Un carácter nulo es escrito inmediatamente después del último carácter leído en el array.
- Si no lee nada (porque no hay caracteres o porque hubo un error) retorna NULL.

FUNCION **fputs**

- **Sintaxis**

`int fputs(const char *cadena, FILE *stream);`

- Escribe la cadena apuntada por **cadena** al stream apuntado por **stream**. El carácter nulo no es escrito.
- *fputs* retorna EOF si ocurre un error de escritura, si no, retorna un valor no negativo.
- `fputs(s, stdout)` equivale a `puts(s)`


```
#include <stdio.h>
int main()
{
    FILE * arch;
    char linea[256];

    arch = fopen("Lectural.txt", "r");
    if (arch==NULL)
        printf("Error al abrir el arch.!\n");
    else {
        fgets(linea, 10, arch); ←
        while (! feof(arch)) {
            fputs(linea, stdout);
            fgets(linea, 10, arch);
        }
        fclose(arch);
    }
    return 0;
}
```

Cuántos
caracteres lee
como máximo?

```
#include <stdio.h>
int main()
{
    FILE * arch;
    char linea[256];

    arch = fopen("Lectural.txt", "r");
    if (arch==NULL)
        printf("Error al abrir el arch.!\n");
    else {
        fgets(linea, 10, arch);
        while (! feof(arch)) {
            fputs(linea, stdout);
            fgets(linea, 10, arch);
        }
        fclose(arch);
    }
    return 0;
}
```

Es necesario
agregar '\0' al
final antes de
imprimir?

```
#include <stdio.h>
int main()
{
    FILE * arch;
    char linea[256];

    arch = fopen("Lectural.txt", "r");
    if (arch==NULL)
        printf("Error al abrir el arch.!\n");
    else {
        fgets(linea, 10, arch);
        while (! feof(arch)) {
            fputs(linea, stdout);
            fgets(linea, 10, arch);
        }
        fclose(arch);
    }
    return 0;
}
```

Dónde imprime?

DESPLAZAMIENTO EN EL ARCHIVO

- Las funciones de lectura y escritura avanzan en el archivo.
- Esta posición de lectura/escritura es relativa al origen del archivo, por eso se llama *desplazamiento*.
- Al abrir un archivo en modo de acceso "r" o "w" el desplazamiento se inicializa en 0 (comienzo del archivo), en cambio, si se utiliza el modo "a" el desplazamiento comienza al final del archivo.

DESPLAZAMIENTO EN EL ARCHIVO

- Es posible obtener el desplazamiento actual de un archivo con la función **ftell**

```
long ftell(FILE *stream);
```

- También es posible modificar el desplazamiento actual de un archivo.
Esto se logra mediante la función **fseek**.

FUNCIÓN **fseek**

- **Sintaxis**

int **fseek**(FILE *stream, long int desplazamiento, int origen);

- Reubica la posición del puntero al archivo.
- La nueva posición, medida en caracteres, es obtenida mediante la suma de **desplazamiento** y la posición especificada por **origen**.
- Los valores para origen son: SEEK_SET (inicio del archivo), SEEK_CUR (actual), SEEK_END (final del archivo)

FUNCIÓN `fseek`

- Suponga que se abre un archivo de 100 bytes de tamaño, en modo de acceso 'r', si a continuación de la apertura se ejecuta:

El nuevo desplazamiento será

- `fseek(fd, 10, SEEK_SET)`

10

- `fseek (fd,-10, SEEK_END)`

90

- `fseek (fd,-10, SEEK_CUR)`

80

```
/* ---- fseek y ftell --- */
#include <stdio.h>
int main()
{ FILE * fd;
  fd=fopen ("prueba.txt", "r");

  if ( fd ) {
    fseek (fd, 0, SEEK_END);
    printf ("El tamaño del archivo es: %u\n",
            ftell(fd));
    fclose (fd);
  }
  else printf("Error al abrir\n");

  return 0;
}
```

Se posiciona al
final del archivo




```
/* ---- fseek y ftell --- */
#include <stdio.h>
int main()
{ FILE * fd;
  fd=fopen ("prueba.txt", "r");

  if ( fd ) {
    fseek (fd, 0, SEEK_END);
    printf ("El tamaño del archivo es: %u\n",
            ftell(fd)); ←
    fclose (fd);
  }
  else printf("Error al abrir\n");

  return 0;
}
```

Retorna el
desplazamiento
actual en bytes

EJERCICIO

- Genere un archivo de texto con información de los empleados de una empresa. Para cada uno se deberá registrar (en una misma línea):
 - Nombre (texto de 10 caracteres)
 - Apellido (texto de 20 caracteres)
 - Sucursal (valor entero)
 - Sueldo (valor real)
- Una vez generado el archivo
 - Vuelva a recorrerlo y visualice en pantalla su contenido.
 - Agregue empleados al final del archivo.