

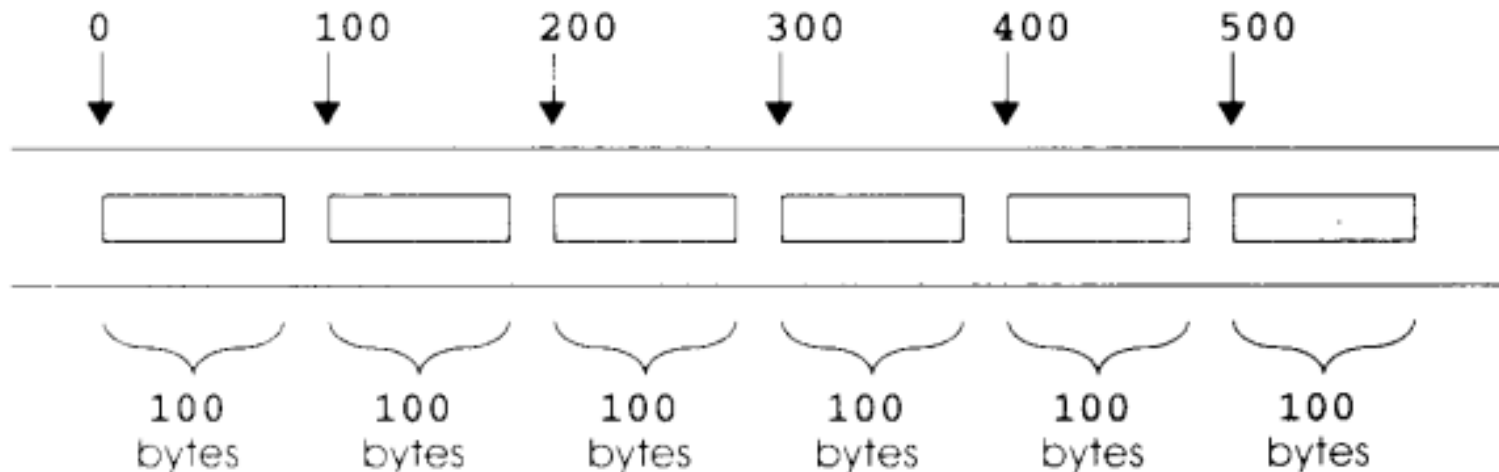


# **Archivos Binarios**

**Acceso Directo**

# Archivos de acceso directo

- Un archivo binario contiene información de cualquier tipo, codificada en forma binaria
- Cada bloque de datos consiste en un número *fijo de bytes continuos*. Esto facilita el manejo de la información.



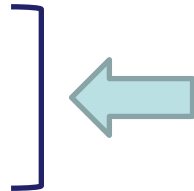
# Modo de apertura de los Archivos Binarios

Modo	Descripción
<b>rb</b>	Abre un archivo binario para lectura
<b>wb</b>	Crea un archivo binario para escritura; si el archivo ya existe se descarta el contenido actual.
<b>ab</b>	Abre o crea un archivo binario para escribir al final del mismo
<b>rb+ ó r+b</b>	Abre un archivo para lectura y escritura.
<b>wb+ ó w+b</b>	Crea un archivo binario para lectura y escritura. Si el archivo existe, se descarta el contenido actual.
<b>ab+ ó a+b</b>	Abre o crea un archivo binario para actualizar. La escritura se realizará al final del archivo.

# Operaciones sobre archivos binarios

- Apertura y cierre

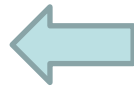
- **fopen**
- **fclose**



Son las mismas que utilizamos para archivos de texto

- Lectura y escritura

- **fread**
- **fwrite**



Veamos como escribir en un archivo binario

- Posicionamiento

- **fseek**

# Creación de un archivo de acceso directo

- La función **fwrite** escribe en un archivo un número especificado de bytes comenzando en una posición de memoria dada.

- En lugar de utilizar:

```
fprintf (arch, "%d", num) ;
```



Escribe de 1 a  
11 bytes

- Utilizamos:

```
fwrite (&num, sizeof (int) , 1, arch) ;
```



Escribe siempre 4 bytes

# Función fwrite

- **Sintaxis**

`size_t fwrite( const void * puntero, size_t tamaño,  
size_t cuantos, FILE * stream);`

- Envía desde el arreglo apuntado por **puntero**, la cantidad de elementos indicada en **cuantos** cuyo tamaño es especificado por **tamaño**, al dispositivo apuntado por **stream**.

# Función **fwrite**

- **Sintaxis**

`size_t fwrite( const void * puntero, size_t tamaño,  
size_t cuantos, FILE * stream);`

- La función ***fwrite*** retorna el número de elementos escritos correctamente, el cual puede ser menor que **cuantos**, pero sólo si se produce un error de escritura.

# Función fwrite

- **Sintaxis**

`size_t fwrite( const void * puntero, size_t tamaño,  
size_t cuantos, FILE * stream);`

- El indicador de posición de ficheros para el stream (si está definido) es avanzado por el número de caracteres escritos correctamente.
- Si existe un error, el valor resultante del indicador de posición de ficheros para el stream es indeterminado.



# Ejemplo

```
#include <stdio.h>
int main()
{
    FILE * arch;
    int valores[10]={3,2,4,1,7,6,5};
    int grabados;

    arch = fopen("AccesoDirecto", "wb");


    if (arch) {
        grabados = fwrite(valores, sizeof(int), 10, arch);

        printf("Se escribieron correctamente %d"
               " numeros enteros\n", grabados);

        fclose(arch);
    }
    return 0;
}
```

**CrearArchBinario.c**

# Lectura de datos de un archivo de acceso directo

- La función **fread** lee un número especificado de bytes de un archivo a la memoria.
- Ejemplo:
  - `fread(vector, sizeof(int), 5, arch);` 

Recibe en **vector** hasta 5 elementos desde el stream apuntado por **arch**. El tamaño de cada uno de estos elementos es la cantidad de bytes de un entero.

# Función fread

- **Sintaxis**

`size_t fread( const void * puntero, size_t tamaño,  
size_t cuantos, FILE * stream);`

- Recibe en el arreglo apuntado por **puntero**, la cantidad de elementos indicada en **cuantos** cuyo tamaño es especificado por **tamaño**, desde dispositivo apuntado por **stream**.

# Función fread

- **Sintaxis**

`size_t fread( const void * puntero, size_t tamaño,  
size_t cuantos, FILE * stream);`

- Retorna el número de caracteres leídos correctamente, el cual puede ser menor que **cuantos** si se encuentra un error de lectura o la marca de fin de archivo.
- El indicador de posición de ficheros para el stream (si está definido) es avanzado por el número de caracteres escritos correctamente.

# Función fread

- **Sintaxis**

`size_t fread( const void * puntero, size_t tamaño,  
size_t cuantos, FILE * stream);`

- Si existe un error, el valor resultante del indicador de posición de ficheros para el stream es indeterminado.
- Si un elemento es parcialmente leído, entonces su valor es indeterminado.

```

#include <stdio.h>
int main()
{
    FILE * arch;
    int v[10], i;

    arch = fopen("AccesoDirecto", "rb");

    if (arch==NULL)
        printf("Error al abrir!");
    else {
        fread(v, sizeof(int), 10, arch);

        for( i=0; i<10; i++)
            printf("v[%d] = %d\n", i, v[i]);

        fclose(arch);
    }
    return 0;
}

```

LeerArchBinario.c

# Función fseek

## Sintaxis

**int fseek(FILE \*arch, long int desplaz, int origen)**

- Modifica el puntero actual del archivo.
- Para un archivo binario, la nueva posición, medida en caracteres desde el principio del fichero, es obtenida mediante la suma de **desplaz** y la posición especificada por **origen** (SEEK\_SET, SEEK\_CUR o SEEK\_END)

# Función fseek

## Sintaxis

**int fseek(FILE \*arch, long int desplaz, int origen)**

- Para un archivo de texto, o bien **desplaz** será cero, o bien **desplaz** será un valor retornado por una llamada anterior a la función *ftell* al mismo archivo y **origen** será **SEEK\_SET**.
- La función *fseek* retorna un valor distinto a cero sólo si una petición no se pudo satisfacer.



```

#include <stdio.h>
int main()
{
    FILE * arch;
    int valor;

    arch = fopen("AccesoDirecto", "rb");

    if (arch==NULL)
        printf("Error al abrir!");
    else {
        fseek(arch, 3*sizeof(int), SEEK_SET);

        fread(&valor, sizeof(int), 1, arch);

        printf("%d\n", valor);

        fclose(arch);
    }
    return 0;
}

```

Leer4toElemento.c

# Ejercicio 1

- Dados 6 números enteros de 8 dígitos cada uno como mínimo (e/8 y 11)
  - Guárdelos en un archivo de texto escribiendo un número en cada línea.
  - Guárdelos en un archivo binario.
- Verifique el tamaño de cada archivo.
- Recupere el 3er número
  - del archivo de texto.
  - del archivo binario.

# Ejercicio 2

- Generar en un archivo binario denominado “inventario.dat” el inventario de una juguetería con el nombre, código (1..9), precio, cantidad vendida y el stock actual (50) de cada juguete. La información se lee desde teclado hasta ingresar un juguete con código 0.
- Para verificar que ha sido creado correctamente, recorra el archivo e imprima su contenido en pantalla.

.

# Ejercicio 3

- El propietario de la juguetería decidió aumentar en un 25% el precio de los juguetes con códigos 2 y 3. Como encargado, debe modificar el inventario almacenado en “inventario.dat” aplicando el aumento indicado.