

ARCHIVOS

INTRODUCCIÓN

Por un lado, conocemos distintos tipos de archivos, según la aplicación que los crea o su función, ejemplos: .docx, .bat, .xlsx, .log, .exe, .dat, .html, .php, .py, .css, etc.

Por otro lado, hasta ahora hemos trabajado con estructuras y datos en memoria y en tiempo de ejecución. El problema que se presenta es que todos los datos procesados y la información obtenida, se eliminan cuando el programa termina.

En ciertos procesos, lo corriente es que se desee utilizar datos que están registrados o también que queden registrados y no desaparezcan cuando el programa finaliza.

Este problema se resuelve utilizando archivos. En términos computacionales y generalizando, un archivo, es una colección de datos que tiene un nombre y una extensión que lo identifica y su contenido puede guardarse en distintos soportes de almacenamiento.

Relacionado con la construcción de distintas aplicaciones que utilizan archivos, un archivo es una corriente o stream, de bytes que posee un final indicado por una marca de fin de archivo. Por qué stream? Las personas que diseñaron C querían una forma uniforme de interactuar con diferentes fuentes de datos, como archivos, tomas, teclados, puertos USB, impresoras, socket, etc, entonces diseñaron una interfaz que podría aplicarse a todos. Esta interfaz utiliza propiedades que son comunes a todos ellos y le dieron un nombre genérico, *streams*.

Y así un stream, como interfaz, permite desentendernos de cómo se gestionan los datos a bajo nivel y a concentrarnos en los objetivos. El identificador de stream en C es una variable tipo FILE* o sea un puntero tipo FILE.

Entonces, para poder acceder a un archivo, se asocia a éste un flujo que lo permite y como se mencionó, un flujo de datos es una abstracción del camino que siguen los datos desde un origen hasta un destino.

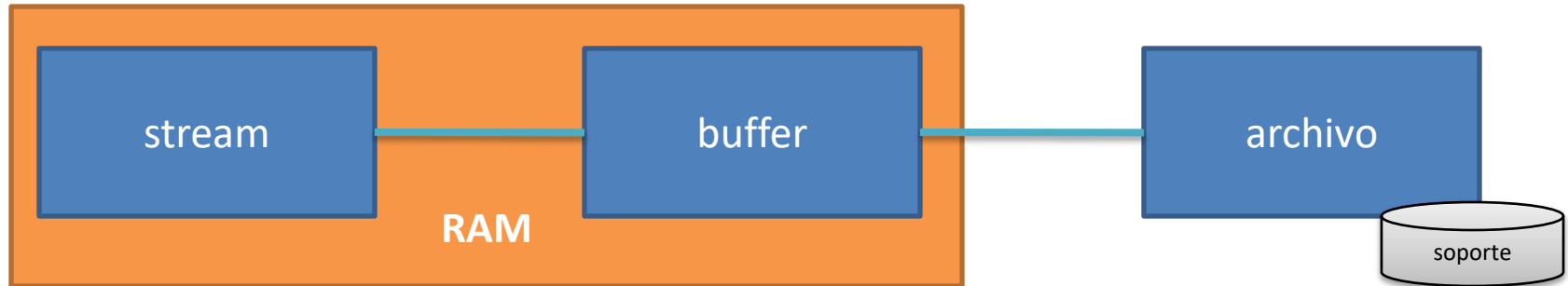
Estos flujos están abiertos por defecto en stdio.h, y nos ofrecen funciones para la entrada y salida estándar:

- Usando los dispositivos estándar (teclado, monitor)
- Usando archivos

Para evitar la diferencia en las operaciones, stdio.h trata a ambos como archivos . Y para representar a cada archivo existen 3 identificadores especiales de tipo FILE *:

- stdin (dispositivo de entrada, teclado)
- stdout (dispositivo de salida, monitor)
- stderr (dispositivo de salida de errores estándar (monitor))

En síntesis, abrir un archivo significa definir un stream. Dicho stream permite la transferencia de datos entre el programa y el archivo en el soporte.



- ✓ El buffer es un área de memoria situada en la RAM asignada al programa que abre el archivo.
- ✓ Toda transferencia de datos entre el programa y el archivo en el soporte se realiza a través del buffer.
- ✓ El acceso a la memoria RAM consume menos tiempo que el acceso a un dispositivo físico.
- ✓ El buffer hace que el número de accesos al archivo físico sea menor.
- ✓ El uso del buffer permite realizar operaciones de entrada salida de forma más eficiente.
- ✓ Los archivos no procesan datos sólo los almacenan.

Clasificación por tipo de los archivos

Archivos de texto

Registros de longitud variable. Pueden ser procesados por cualquier editor de texto . Contienen información en forma de caracteres. Normalmente se organizan los caracteres en forma de líneas al final de cada cual se coloca un carácter de fin de línea (normalmente “\r\n”).

Archivo binarios

Registros de longitud fija. Almacenan datos en forma binaria, es decir que no son interpretables como texto (números, imágenes, etc.).

Clasificación por la forma de acceso

Según la forma en la que accedamos a los archivos disponemos de dos tipos de archivo:

Archivos de acceso secuencial

Como se trata de archivos en los que el contenido se lee o escribe de forma continua no se puede acceder a un punto concreto del archivo, es decir que para leer cualquier información necesitamos leer todos los datos hasta llegar a dicha información, o sea que para leer el registro n hay que leer los n-1 registros anteriores. En general son los archivos de texto los que utilizan en el acceso de forma secuencial.

Archivos de acceso aleatorio o acceso directo

Se puede acceder a cualquier dato del archivo conociendo su posición en el mismo, es decir que se puede acceder a un registro concreto sin necesidad de leer todos los anteriores. Dicha posición se suele indicar en bytes.
En general sin los archivos binarios los que se utilizan en el acceso directo.

Estructura FILE y punteros a archivos

En el archivo de cabecera stdio.h se define una estructura llamada FILE. Esa estructura representa la cabecera de los archivos. La secuencia de acceso a un archivo debe poseer esta estructura. Un programa requiere tener un puntero de tipo FILE* por cada archivo que se desee leer o escribir. A este puntero se le llama puntero de archivos.

Operaciones con archivos

Los flujos de E/S solo pueden transferir datos en una dirección, esto significa que se tienen que definir flujos diferentes para lectura y escritura de datos.

- ✓ Abrir archivo
- ✓ Escribir archivo
- ✓ Leer archivo
- ✓ Modificar archivo
- ✓ Emitir o listar contenido de archivo
- ✓ Cerrar archivo
- ✓ Funciones de control y procesos

ARCHIVOS DE TEXTO

PRIMERA PARTE

Habíamos mencionado que un archivo de texto es una secuencia de caracteres organizadas en líneas terminadas por un carácter de nueva línea. En estos archivos se pueden almacenar canciones, código fuente de programas, frases, etc. Es decir que son “entendibles” .

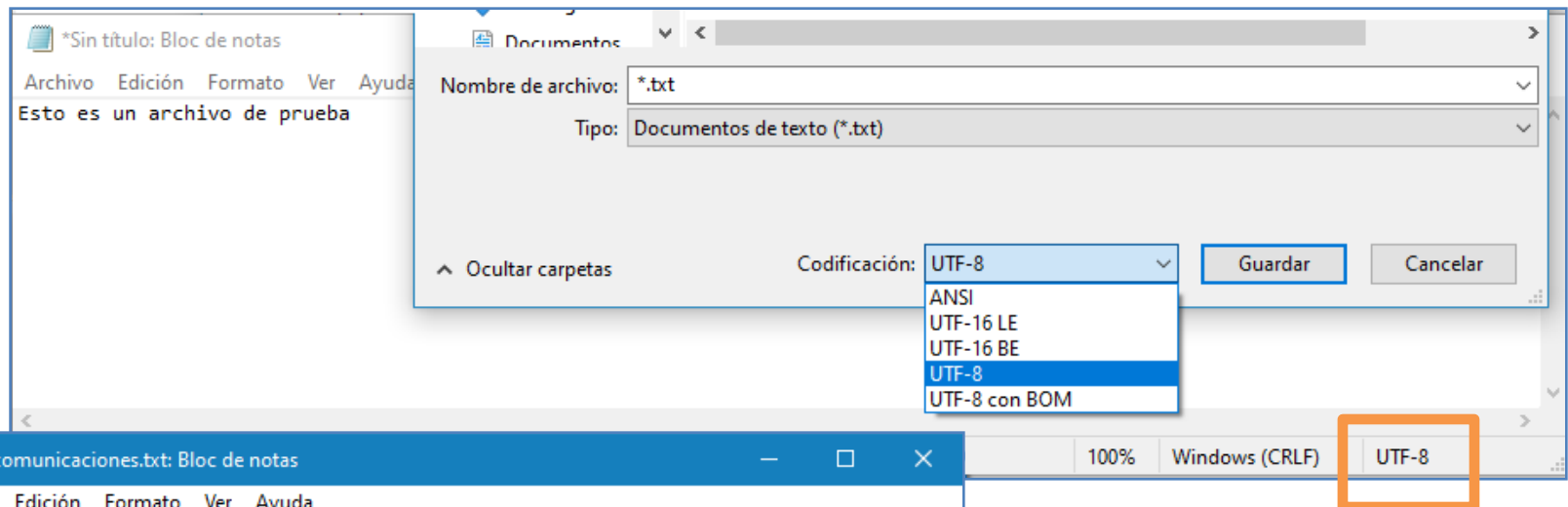
Los datos en un archivo de texto se almacenan usando el código ASCII, en el cual cada carácter es representado por un byte.

Por lo tanto, todos sus datos se almacenan como cadenas de caracteres, es decir, los números se almacenan con su representación ASCII y NO su representación numérica, por lo tanto NO se pueden realizar operaciones matemáticas directamente con ellos, por ejemplo si se guarda el dato 3.141592 en un archivo de texto, se almacena como “3.141592” y nótese que ... $3.141592 \neq \text{“3.141592”}$

Los archivos de texto se caracterizan por ser planos, es decir, todas las letras tienen el mismo formato y no hay palabras subrayadas, en negrita, o letras de distinto tamaño o ancho.

Al leerlos hay que tener en cuenta la que la codificación de caracteres puede variar , la 'ñ' se puede codificar muy distinto según qué sistema utilicemos, vemos algunas codificaciones:

- ASCII:** Código de 7 bits que permite incluir 128 caracteres. Pero no están los caracteres nacionales por ejemplo la 'ñ' del español, ni otros símbolos. Por ello se usó el octavo bit para producir códigos de 8 bits, llamado ASCII extendido (pero los ASCII de 8 bits son diferentes en cada país).
- ISO 8859-1:** El más usado en occidente. Se la llama codificación de Europa Occidental. Son 8 bits con el código ASCII más los símbolos frecuentes del inglés, español, francés, italiano o alemán entre otras lenguas de Europa Occidental.
- Windows 1252:** Windows llama ANSI a esta codificación. En realidad se trata de un superconjunto de ISO 8859-1 que es utilizado en el almacenamiento de texto por parte de Windows.
- Unicode:** La norma que intenta unificar criterios para hacer compatible la lectura de caracteres en cualquier idioma. Hay varias posibilidades de aplicación de Unicode, la más utilizada en la actualidad es la UTF-8 que es totalmente compatible con ASCII



Funciones de manejo de archivos

fopen() Abre un archivo.

fclose() Cierra un archivo.

fseek() Busca un byte específico de un archivo.

feof() Devuelve cierto si se llega al final del archivo.

ferror() Devuelve cierto si se produce un error.

rewind() Coloca el localizador de posición del archivo al principio del mismo.

remove() Borra un archivo.

rename() renombra un archivo

fflush() Vacía un archivo.

Funciones de E/S en streams de texto:

E/S de caracteres:

fgetc() Lee 1 carácter

fputc() Escribe 1 carácter

E/S de cadenas:

fgets() Lee 1 línea

fputs() Escribe 1 línea

E/S con formato:

fscanf() Lee con formato

fprintf() Escribe con formato

Modos de apertura en archivos de texto	Operación
r ó rt	Apertura en modo sólo lectura, el archivo debe existir
w ó wt	Apertura en modo escritura, si el archivo existe se sobrescribe y pierde el contenido anterior. Si no existe lo crea.
a ó at	Apertura en modo agregar, si el archivo existe agrega los datos al final del archivo, si no existe lo crea.
r+	Apertura en modo lectura / escritura, el archivo debe existir.
w+	Apertura en modo lectura / escritura, el archivo debe existir, si el archivo existe se sobrescribe y pierde el contenido anterior. Si no existe lo crea.
a+	Apertura en modo lectura / agregar, si el archivo existe agrega los datos al final del archivo, si no existe lo crea.

```
// ejemplo 1
#include <stdio.h>
```

fopen(<nombre de archivo>,<modificador>)

```
int main(){
char caracter;
FILE * pArchivo;

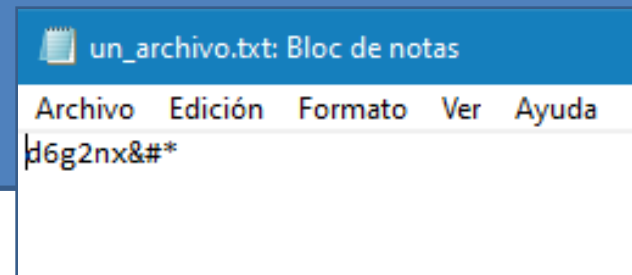
pArchivo = fopen("un_archivo.txt","w");

if(pArchivo != NULL){
    do{
        printf("Ingrese un caracter: ");
        scanf("%c",&caracter);
        fflush(stdin);
        fputc(caracter, pArchivo);
    }while(caracter != '*');

    fclose(pArchivo);}
else printf("Error en la apertura del archivo!");

getchar();
return 0;
}
```

```
Ingrese un caracter: d
Ingrese un caracter: 6
Ingrese un caracter: g
Ingrese un caracter: 2
Ingrese un caracter: n
Ingrese un caracter: x
Ingrese un caracter: &
Ingrese un caracter: #
Ingrese un caracter: *
```



```
//ejemplo 2
#include <stdio.h>

int main(){
char palabra[30];
FILE * pArchivo;

pArchivo = fopen("un_archivo.txt","w");

if(pArchivo != NULL){
    printf("Ingrese una oración hasta 50 caracteres: \n");
    scanf("%s",palabra);
    fflush(stdin);
    fputs(palabra, pArchivo);
    fclose(pArchivo);
}
else printf("Error en la apertura del archivo!");

getchar();
return 0;
}
```

fputs() escribe palabras en el archivo

```
Ingrese una palabra:
laboratorio
```

un_archivo.txt: Bloc de notas

Archivo	Edición	Formato	Ver	Ayuda
laboratorio				

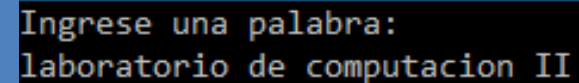
```
//ejemplo 3
#include <stdio.h>

int main(){
char palabra[30];
FILE * pArchivo;

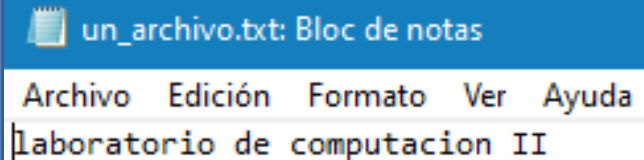
pArchivo = fopen("un_archivo.txt","w");

if(pArchivo != NULL){
    printf("Ingrese una oración hasta 50 caracteres: \n");
    gets(palabra);
    fflush(stdin);
    fputs(palabra, pArchivo);
    fclose(pArchivo);
}
else printf("Error en la apertura del archivo!");

getchar();
return 0;
}
```



Ingrese una palabra:
laboratorio de computacion II



un_archivo.txt: Bloc de notas

Archivo	Edición	Formato	Ver	Ayuda
---------	---------	---------	-----	-------

laboratorio de computacion II

```
// ejemplo 4
#include <stdio.h>
```

```
int main(){
int n;
char nombre[30];
float salario;
```

```
FILE *pArchivo;
```

```
pArchivo = fopen("empleados.txt", "w");
```

```
if(pArchivo!=NULL){
do{
```

```
printf("Ingrese el número de legajo del empleado\nPara terminar ingrese 0 o número menor que 0:\n",
163,163);
```

```
scanf("%d",&n);
```

```
if(n>0){
```

```
printf("Ingrese el nombre del empleado: ");
```

```
scanf("%s",nombre);
```

```
printf("Ingrese el salario del empleado: ");
```

```
scanf("%f",&salario);
```

```
fprintf(pArchivo,"%d%15s%15f\n", n,nombre,salario);
```

```
}
```

```
} while(n>0);
```


```
close(pArchivo);}
```

```
else{ printf("Error en la apertura del archivo!");}
```

```
getchar();
```

```
return 0;}
```

```
Ingrese el nombre del empleado: Pepe
Ingrese el salario del empleado: 34000
Ingrese el número de legajo del empleado
Para terminar ingrese 0 o número menor que 0:
5
Ingrese el nombre del empleado: Lila
Ingrese el salario del empleado: 35000
Ingrese el número de legajo del empleado
Para terminar ingrese 0 o número menor que 0:
2
Ingrese el nombre del empleado: Lucho
Ingrese el salario del empleado: 32000
Ingrese el número de legajo del empleado
Para terminar ingrese 0 o número menor que 0:
0
```

 empleados.txt: Bloc de notas

Archivo	Edición	Formato	Ver	Ayuda
3	Pepe	34000.000000		
5	Lila	35000.000000		
2	Lucho	32000.000000		

fprintf() escribe los datos con formato en el archivo


```
// ejemplo 5
#include <stdio.h>

int main(){
    int n;
    char nombre[30];
    float salario;

    FILE *pArchivo;
    pArchivo = fopen("empleados.txt","a");
    if(pArchivo!=NULL){
        do{
            printf("Ingrese el n°mero de legajo del empleado\nPara terminar ingrese 0 o n°mero menor que 0:\n",
163,163);
            scanf("%d",&n);
            if(n>0){
                printf("Ingrese el nombre del empleado: ");
                scanf("%s",nombre);
                printf("Ingrese el salario del empleado: ");
                scanf("%f",&salario);
                fprintf(pArchivo,"%d%15s%15f\n", n,nombre,salario);
            }
        } while(n>0);
        close(pArchivo);
    } else{ printf("Error en la apertura del archivo!");}
    getchar();
    return 0;}

```

```

Ingrese el número de legajo del empleado
Para terminar ingrese 0 o número menor que 0:
4
Ingrese el nombre del empleado: Pachi
Ingrese el salario del empleado: 34500
Ingrese el número de legajo del empleado
Para terminar ingrese 0 o número menor que 0:
7
Ingrese el nombre del empleado: Gala
Ingrese el salario del empleado: 36000
Ingrese el número de legajo del empleado
Para terminar ingrese 0 o número menor que 0:

```

empleados.txt: Bloc de notas				
Archivo	Edición	Formato	Ver	Ayuda
3	Pepe	34000.000000		
5	Lila	35000.000000		
2	Lucho	32000.000000		
4	Pachi	34500.000000		
7	Gala	36000.000000		

```
#include <stdio.h> //ejemplo 6
```

```
#include <stdlib.h>
```

```
void crearArch(FILE * pA, char * nom){
```

```
    pA = fopen(nom,"w");
```

```
    if (pA != NULL){
```

```
        printf("Archivo creado exitosamente.\n");
```

```
        fclose(pA); }
```

```
    else { printf("Error en la creación del archivo!"); } }
```

```
void escribirArch(FILE * pA, char * nom){
```

```
    int n;
```

```
    char nombre[30];
```

```
    float salario;
```

```
    pA = fopen(nom,"a");
```

```
    if(pA!=NULL){
```

```
        do{
```

```
            printf("Ingrese el n°mero de legajo del  
empleado\nPara terminar ingrese 0 o n°mero menor  
que 0:\n", 163,163);
```

```
            scanf("%d",&n);
```

```
            if(n>0){
```

```
                printf("Ingrese el nombre del empleado: ");
```

```
                scanf("%s",nombre);
```

```
                printf("Ingrese el salario del empleado: ");
```

```
                scanf("%f",&salario);
```

```
                fprintf(pA,"%d%15s%15f\n", n,nombre,salario);
```

```
            } } while(n>0);
```

```
    fclose(pA); }
```

```
    else{ printf("Error en la apertura del archivo!"); } }
```

```
int main(){
```

```
    FILE * pArchivo;
```

```
    char nombreArch[30] = "empleados.txt";
```

```
    int opcion;
```

```
    printf("Elija la operaci%cn a realizar\n", 162);
```

```
    printf("1 - Crear Archivo\n");
```

```
    printf("2 - Agregar datos al Archivo\n");
```

```
    printf("3 - Salir del programa\n");
```

```
    scanf("%d",&opcion);
```

```
    switch(opcion){
```

```
        case 1: crearArch(pArchivo, nombreArch);
```

```
            break;
```

```
        case 2: escribirArch(pArchivo, nombreArch);
```

```
            break;
```

```
        case 3: exit(1);
```

```
        default: printf("Ingreso un valor inv%clido, fin del  
programa...\n", 160);
```

```
    }
```

```
    getchar();
```

```
    return 0;}
```

empleados.txt: Bloc de notas				
Archivo	Edición	Formato	Ver	Ayuda
3	Pepe	34000.000000		
5	Lila	35000.000000		
2	Lucho	32000.000000		
4	Pachi	34500.000000		
7	Gala	36000.000000		
9	Carli	35000.000000		
10	Pancho	30000.000000		

```
Elija la operación a realizar
1 - Crear Archivo
2 - Agregar datos al Archivo
3 - Salir del programa
2
Ingrese el número de legajo del empleado
Para terminar ingrese 0 o número menor que 0:
9
Ingrese el nombre del empleado: Carli
Ingrese el salario del empleado: 35000
Ingrese el número de legajo del empleado
Para terminar ingrese 0 o número menor que 0:
10
Ingrese el nombre del empleado: Pancho
Ingrese el salario del empleado: 30000
Ingrese el número de legajo del empleado
Para terminar ingrese 0 o número menor que 0:
0
```

```
// ejemplo 7
#include <stdio.h>

void leerArch(FILE * pA, char * nom){
    char c;
    pA=fopen(nom,"r");
    if(pA != NULL){
        while( c != EOF) {
            c = fgetc(pA);
            printf("%c", c);
        }
        fclose(pA);
    }
    else { printf("Error en la apertura del archivo!");}
}

int main(){
    FILE * pArchivo;
    char nombreArch[30] = "un_texto.xyz";
    leerArch(pArchivo, nombreArch);
    getchar();
    return 0;
}
```

EOF (end-of-file) indica que no hay más información en el archivo

La realidad física de los datos es que éstos son n^o meros binarios. Como es prácticamente imposible trabajar utilizando el código binario, los datos deben de ser reinterpretados como enteros, caracteres, cadenas, estructuras, etc. Al leer un archivo los datos de éste pueden ser leídos como si fueran binarios, o utilizando otra estructura más apropiada para su lectura por parte del programador. A esas estructuras se les llama registros y equivalen a las estructuras (structs) del lenguaje C. Un archivo así entendido es una colección de registros que poseen la misma estructura interna. Cada registro se compone de una serie de campos que pueden ser de tipos distintos (incluso un campo podría ser una estructura o un array). En cada campo los datos se pueden leer según el tipo de datos que almacenen (enteros, caracteres,...), pero en realidad son unos y ceros.

```
// ejemplo 8
#include <stdio.h>

void leerArch(FILE * pA, char * nom){
    char c;
    pA=fopen(nom,"r");
    if(pA != NULL){
        while( ! feof ( pA ) ) {
            c = fgetc(pA);
            printf("%c", c);
        }
        fclose(pA);
    }
    else { printf("Error en la apertura del archivo!");}
}

int main(){
    FILE * pArchivo;
    char nombreArch[30] = "un_texto.xyz";
    leerArch(pArchivo, nombreArch);
    getchar();
    return 0;
}
```

!feof() permite el recorrido secuencial del archivo hasta que encuentre el indicador de final del mismo.

La realidad física de los datos es que éstos son n^o meros binarios. Como es prácticamente imposible trabajar utilizando el código binario, los datos deben de ser reinterpretados como enteros, caracteres, cadenas, estructuras, etc. Al leer un archivo los datos de éste pueden ser leídos como si fueran binarios, o utilizando otra estructura más apropiada para su lectura por parte del programador. A esas estructuras se les llama registros y equivalen a las estructuras (structs) del lenguaje C. Un archivo así entendido es una colección de registros que poseen la misma estructura interna. Cada registro se compone de una serie de campos que pueden ser de tipos distintos (incluso un campo podría ser una estructura o un array). En cada campo los datos se pueden leer según el tipo de datos que almacenen (enteros, caracteres,...), pero en realidad son unos y ceros.

ejemplo 9

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

```
void leerArch(FILE * pA, char * nom){
```

```
    char linea[200];
```

```
    pA=fopen(nom,"r");
```

```
    if(pA != NULL){
```

```
        fgets(linea,200,pA); //carga una línea
```

```
        while(!feof(pA)){
```

```
            printf("%s", linea);
```

```
            fgets(linea,200,pA);
```

```
        }
```

```
        fclose(pA);
```

```
    }
```

```
    else { printf("Error en la apertura del archivo!");}
```

```
}
```

```
int main(){
```

```
    FILE * pArchivo;
```

```
    char nombreArch[30] = "un_texto.xyz";
```

```
    leerArch(pArchivo, nombreArch);
```

```
    getchar();
```

```
    return 0;
```

```
}
```

fgets() obtiene líneas de texto del archivo que, en este caso, emitimos.

La realidad física de los datos es que éstos son n^o meros binarios. Como es prácticamente imposible trabajar utilizando el código binario, los datos deben de ser reinterpretados como enteros, caracteres, cadenas, estructuras, etc. Al leer un archivo los datos de éste pueden ser leídos como si fueran binarios, o utilizando otra estructura más apropiada para su lectura por parte del programador. A esas estructuras se les llama registros y equivalen a las estructuras (structs) del lenguaje C. Un archivo así entendido es una colección de registros que poseen la misma estructura interna. Cada registro se compone de una serie de campos que pueden ser de tipos distintos (incluso un campo podría ser una estructura o un array). En cada campo los datos se pueden leer según el tipo de datos que almacenen (enteros, caracteres,...), pero en realidad son unos y ceros.

```

# ejemplo 10
#include <stdio.h>

void leerArch(FILE * pA, char * nom){
    int n;
    char nombre[30];
    float salario;
    pA = fopen(nom,"r");

    if(pA!=NULL){
        while(!feof(pA)){
            fscanf(pA,"%d%15s%15f\n", &n,nombre,&salario);
            printf("%d\t%s\t%.2lf\n",n,nombre,salario);
        }
        fclose(pA);
    }
    else{ printf("Error en la apertura del archivo!");}
}

int main(){
    FILE * pArchivo;
    char nombreArch[30] = "empleados.txt";
    leerArch(pArchivo, nombreArch);
    getchar();
    return 0;}

```

fscanf () lee los datos con formato del archivo.

```

3      Pepe      34000.00
5      Lila      35000.00
2      Lucho     32000.00
4      Pachi     34500.00
7      Gala      36000.00
9      Carli     35000.00
10     Pancho    30000.00

```

CONTINUARÁ...