

## Asignatura Programación

### Apuntes De Clase

### Archivos En Disco

#### Introducción

La manera de capturar la información y mostrar los resultados producidos por los programas usada hasta ahora fue a través de la entrada y salida estándar (teclado y monitor); así el almacenamiento de datos y resultados usado tiene carácter temporal, al terminar un programa todos los datos se pierden, pues se almacenan en memoria RAM, que es del tipo volátil.

A partir de ahora, Ud. podrá escribir programas cuya entrada y salida no estén conectadas al programa y para conservar los datos y resultados, usará archivos, en los cuales podrá almacenar grandes volúmenes de información.

#### Terminología Básica

Un archivo es una secuencia de elementos de información que se almacena en algún medio de escritura que permita ser leído o accedido por un programa. Un archivo es identificado por un nombre y eventualmente por ruta de acceso al dispositivo que lo contiene.

Para el sistema operativo, en la mayoría de los casos, un archivo es un flujo o secuencia de bytes que es tratado como una única unidad lógica. Un archivo de datos tiene un tamaño, que generalmente se expresa en bytes y un nombre

La información de un archivo consiste en “paquetes” pequeños de información, llamados registros, que son individualmente diferentes pero que comparten algún rasgo en común. La manera en que se agrupa la información en un archivo depende completamente de la persona que diseñe el archivo. Los programas que acceden a los archivos pueden crear, modificar y borrar archivos. Los programadores que crean los programas deciden qué archivos necesitan, cómo se van a usar y (a menudo) sus nombres.

Por ejemplo, en un sistema de software que maneje la información de los animales de un zoológico, un registro para un animal en particular, puede estar formado por los siguientes campos:

- Número de jaula entero
- Especie del animal cadena de caracteres
- Nombre del animal cadena de caracteres
- Fecha de nacimiento fecha
- Peso real
- Kilogramos de comida diaria real
- Frecuencia de limpieza de jaula entero // cantidad de veces por día
- Estado de la salud del animal carácter // B: buena, M: mala, R: regular
- Descendencia si ó no
- Peligroso si ó no

En el ejemplo, cada uno de los campos corresponde a un animal; si se trata de un zoológico, el sistema de software tendrá que almacenar la información de muchos animales y para cada uno de ellos tendrá que crear un registro y guardar allí la información particular de cada uno. Todos estos registros conformarán un tipo particular de archivo.

#### Flujos De Bytes Y Archivos En Disco

El lenguaje C considera a cada uno de los archivos como un flujo de bytes. Cada archivo termina con un marcador de fin de archivo.

Cuando un archivo se abre se establecen canales de comunicación (flujos) entre el programa y archivos, se asocia un flujo con el archivo. Se puede expresar que, cuando un programa empieza a funcionar, se abren automáticamente 3 archivos y sus flujos asociados:

- La entrada estándar: el flujo de entrada permite que el programa lea datos del teclado.

- La salida estándar: el flujo de salida estándar permite que el programa imprima datos por la pantalla.
- El error estándar: se ocupa de enviar mensajes de error al usuario a través de la pantalla.

Estos archivos pertenecen a la biblioteca <stdio.h >.

Para trabajar con programas con entrada y salida no estándares, se usarán archivos en disco.

### De La Organización Física Y Del Acceso A Los Archivos

Para el lenguaje C existen dos tipos diferentes de archivos de datos: archivos secuenciales de datos (o estándar) y archivos orientados al sistema (de bajo nivel). Generalmente se trabaja con archivos de datos secuenciales.

Ventajas de la organización secuencial:

- Es la más sencilla de manejar para el programador.
- Si hay que acceder a un conjunto de registros consecutivos, o a todo el archivo, es el método más rápido.
- No deja espacios entre registro y registro, por lo que se optimiza el uso del espacio en la memoria secundaria.

Inconvenientes:

- Para consultar datos individuales, hay que recorrer todo el archivo desde el principio. Es decir, el acceso a registros individuales es, en general, lento.
- Las operaciones de inserción y eliminación de registros solo pueden hacerse al final del archivo. Hacerlas con registros intermedios representa mover grandes bloques de información y, por lo tanto, consumir mucho tiempo.

El tipo de acceso al archivo es el procedimiento que se usa para acceder a un registro concreto y efectuar algún tipo de transacción con él.

Un acceso relativo permite acceder a los archivos velozmente, se puede llegar a la información buscada en forma directa.

Un acceso secuencial es más lento. De esta manera, el acceso a la información se hace en el orden en el cual fue grabada, o bien, si el archivo es de escritura, la información se grabará en forma secuencial o sucesiva. Pareciera que el acceso secuencial es limitante, pero muchas aplicaciones se adaptan muy bien a este procesamiento

Un acceso relativo es más versátil que un acceso secuencial, porque permite acceder a cualquier parte del fichero en cualquier momento, como si fueran arrays. Las operaciones de lectura y escritura pueden hacerse en cualquier punto del archivo.

En este curso de Programación, se usarán archivos de organización y acceso secuencial.

### Dos Tipos Distintos De Archivos

Un archivo de texto es un archivo ASCII compatible la mayor parte de los lenguajes de programación y sus aplicaciones. Estos archivos no siempre contienen texto, en el sentido de un procesador de textos, puede contener líneas de texto, correspondientes a líneas impresas en una hoja de papel. Todo dato puede ser almacenado como texto.

Si no indico de qué tipo de archivo se trata, por default el sistema lo interpreta como "archivo de texto", o sea un archivo conformado por líneas de caracteres que terminan con el carácter "\n".

Cuando se usan archivos de texto ocurren ciertas traducciones automáticas, por sobre todo de caracteres de control y puede ocurrir que la cantidad de caracteres difiera con la cantidad original ingresada.

Un archivo de texto puede contener líneas de texto, correspondientes a líneas impresas en una hoja de papel. En algunos casos, los programas de computadora manipulan los archivos que se hacen visibles al usuario de la computadora. En el contexto de un programa el usuario puede manipular sus propios archivos de texto. El contenido del archivo está organizado según la decisión del programador y

cualquier programa de procesamiento de texto lo puede acceder. El usuario elige el nombre y la ubicación del archivo y proporciona la información (como palabras y texto) que se almacenará en el archivo.

Un archivo binario es un archivo que está comprimido. Los datos ocupan menos espacio que en los archivos de texto. En un archivo binario la secuencia de bytes que los componen no están procesados, los datos están en formato interno y no tienen ningún tipo de traducción. Los archivos binarios solo pueden ser accedidos por los programas en código C escritos para y por ellos.

No todas las computadoras pueden leer un archivo binario creado por otra computadora, lo cual conspira contra la portabilidad del programa.

¿Cómo se decide qué tipo de archivo usar? ¿Archivo de texto o archivo binario? Depende de la finalidad del archivo.

Si el archivo sirve para guardar información que luego habrá de ser procesada por otros programas, y si la misma no debe ser “imprimible”, se elige trabajar con archivos de tipo binario, que son archivos no editables.

En cambio, si la propuesta es que la información pueda ser editada, impresa, o listada, deberá elegirse trabajar con archivos de modo texto.

En ambos casos, la forma de crear los archivos es idéntica, lo que cambia es la elección de las funciones para leer y escribir.

Por ejemplo, si el sistema de software que maneja la información del zoológico debe emitir un listado de la información de los datos de mantenimiento de la jaula de cada animal, o la información de los datos de cada animal o los valores del costo de mantenimiento mensual de las jaulas, el programador seguramente escogerá trabajar con archivos de texto.

### Para Trabajar Con Un Archivo

Desde un punto de vista funcional las operaciones que se pueden llevar a cabo con archivos son: apertura, cierre, escritura, lectura, entre otras.

Para usar un archivo es necesario usar una variable de tipo puntero llamada Puntero de Archivo, cuyo tipo asociado es el tipo predefinido FILE, tipo estructurado que contiene información de tipo estadística del archivo y está definida en <stdio.h>. En consecuencia, los archivos se declaran como una variable en los programas.

La forma general de declarar un archivo es:

```
FILE *nombre_de_archi;
```

Por ejemplo: FILE \*JAULAS;

En el marco de un programa, el código hace referencia al archivo por el nombre, para indicar que operación habrá de llevar a cabo con el mismo, pero es el sistema operativo y el lenguaje los que se encargan de ubicarlo.

Respecto del nombre del archivo, se puede usar el mismo identificador del archivo tanto en el medio interno (programa) como en el medio externo (dispositivo donde está grabado),

### Apertura De Un Archivo

Cuando Ud. va a abrir un archivo en disco debe indicar cuál es el nombre y para que va a usarlo. Las operaciones que puede llevar a cabo: Grabar, Escribir, Agregar, Leer, entre otras.

El lenguaje C y el sistema operativo llevan a cabo algunas transacciones para que ellas se puedan realizar. Para la apertura se usará la función fopen definida en la biblioteca <stdio.h>

El prototipo de fopen es: FILE \*fopen (char \*nombre, char \*modo);

Donde:

char \*nombre, es una cadena de caracteres que guarda el nombre del archivo y si es

necesario indica la ruta de acceso al mismo.

char \*modo, es una cadena de caracteres que indica cómo se puede usar el archivo.

donde nombre es el nombre del archivo y modo es un carácter que indica el modo en el que el fichero se desea abrir:

```
FILE *pf; // declaración de un puntero a archivo
pf = fopen("nombre", "modo");
```

La función devuelve un puntero que apunta a un archivo. Los modos de apertura de un archivo pueden ser, entre otras: "r", "w", "a", lectura, escritura desde el comienzo del archivo y añadido al final del archivo respectivamente.

También puede ser :rb sólo lectura (archivo binario), wb escritura desde el comienzo del archivo (archivo binario) y ab escritura añadida al final del archivo (archivo binario).

```
Forma de uso:      FILE *archi;
                  archi= fopen("archi.dat","r");
```

Se suelen ver programas que agregan una "t" o una "b" a los modos de apertura, por ejemplo:

```
FILE *archivo;
Archi= fopen("archiv.dat","rb");
```

La "t" indica que se está trabajando con un archivo de texto, en tanto la "b" indica que el archivo es binario.

El puntero pf devuelto por fopen() será NULL (=0) si por alguna razón no se ha conseguido abrir el fichero en la forma deseada.

Estos ejemplos valen cuando el archivo se encuentra en el directorio de trabajo sobre el cual se está trabajando, caso contrario se debe indicar la ruta de acceso. Si el archivo está en el disco G, se escribirá:

```
Archi= fopen("G:\\archiv.dat","rb");
```

Otra forma de uso: usando una rutina de detección de errores

Durante la operación de apertura puede ocurrir algún error, en cuyo caso el lenguaje retornará un valor que equivale a NULL. Para evitar estas situaciones, se recomienda usar rutinas de detección de errores que permiten detectar si la operación que se lleva a cabo con el archivo: apertura o cierre terminó en forma exitosa.

```
if ((archi = fopen("c:\\zoo.bin", "rb"))== NULL)
{
    printf("imposible abrir el archivo\n");
    return 1;    // exit(0);
}
```

Recuerde que la proposición exit termina la ejecución del programa. Una convención establece que: exit(0) indica una salida normal del programa, en tanto exit(k) una salida anormal, donde k responde a códigos de errores preestablecidos.

La proposición exit invoca adecuadamente a la función de cierre fclose, para cerrar todos los archivos circulantes y evitar accidentes.

### Cierre De Un Archivo

Todos los archivos abiertos con fopen durante la ejecución de un programa deben cerrarse con la función inversa: fclose. Esta función cierra formalmente los archivos a nivel de sistema operativo, en realidad lo que hace es interrumpir la conexión que fue establecida por fopen entre el puntero de archivo y el nombre externo, liberando al puntero de archivo.

Para el cierre se usará la función fclose definida en la biblioteca <stdio.h>

El prototipo de fclose es: `int fclose (FILE *nombre_arch);`

```
Forma de uso:      FILE *archi;
                  archi= fopen ("archi.dat","r");
                  :
                  :
                  fclose(archi);
```

El fallo del cierre provoca problemas tales como la pérdida de información, archivos dañados o corruptos.

## Desde Un Punto De Vista Operativo

### - Para La Entrada Y Salida De Caracteres

#### E/S Interactiva

```
char car;
:
car = getchar();      // lectura interactiva
:
putchar(car);        // escritura interactiva
```

#### E/S por Archivos

```
FILE *archi1,archi2;
archi1= fopen("archi1","r");      //archivos de texto
archi2=fopen("archi2","w");

// rutina de detección de errores
:
car=fgetc(archi1);      // lectura desde archivo
fputc(archi,car); // escritura en el archivo
                        // después de las lecturas y escrituras
fclose(archi1);
fclose(archi2);
```

### End Of File

El lenguaje C indica el fin de los datos de un archivo con un valor distintivo. Este valor se llama EOF por End Of File (Fin de archivo). Es un valor definido en <stdio.h>.

El carácter **EOF** existe en la tabla ASCII aunque no es este el carácter que indica el fin-de-archivo en la fuente de datos, ya que un archivo no contiene como último byte el carácter **EOF**.

En la librería estándar de C, el acceso a un fichero y otras funciones I/O pueden devolver un valor igual al valor **EOF** para indicar que la condición end-of-file se ha cumplido, lo cual lo hace muy apropiado para el tratamiento de los archivos de organización secuencial.

### - Para La Entrada Y Salida De Cadenas

#### E/S Interactiva

```
#define MAXIMO 30
:
char cadena[MAXIMO];
:
gets( cadena);
puts(cadena);
```

E/S por Archivos

```
FILE *archi1,archi2;
archi1= fopen("archi1","r");           // Archivos de texto
archi2= fopen("archi2","w");
:
:
fgets(cadena, MAXIMO, archi1);
fputs(cadena, archi2);
```

**- Para La Entrada Y Salida Con Formato**E/S Interactiva

```
int i;
:
printf("Numeros Cuadrados\n");
printf("\n");
for (i=1;i<=20;i++)
    printf(" %3d\n",i);

:
```

E/S por Archivos

```
FILE *cuad;
int i;
:
cuad = fopen("cuad.dat","w");           // Archivos de texto
:
fprintf(cuad,"Numeros Cuadrados\n");
fprintf(cuad,"\n");
for (i=1;i<=20;i++)
    fprintf(cuad," %3d \n",i);
:
fclose(cuad);
```

De manera similar se usará la función de entrada con formato de archivo, fscanff.

**Cuando Se Tiene Que Elaborar Aplicaciones Importantes**

En general, los programadores leen y escriben registros de un archivo. Un registro de archivo es una colección de uno o más valores (campos) que el programador lee o escribe en bloque en el disco. La idea es semejante a lo que se conoce como estructuras (ver título **Terminología Básica**).

A diferencia de otros lenguajes de programación, el C no impone formas estructuradas fijas para un archivo. Es el programador quien debe definir alguna estructura de registro de archivo e imponerla en su aplicación.

Los archivos de acceso secuencial son el tipo de organización más usada para registros de un archivo. Con este modelo se tiene acceso a los registros en forma consecutiva hasta que son localizados los datos deseados y si el modo de acceso se declara como binario, mejora notablemente el rendimiento de la aplicación.

Para leer y escribir registros de una archivo, se usarán dos funciones: fread y fwrite de la biblioteca <stdio.h>. Estas funciones permiten trabajar con bloques de datos, desde atómicos hasta registros, pasando por los arreglos u otros tipos de datos más complejos.

Para usarlas, es necesario conocer el operador sizeof.

**El operador sizeof**

Este operador produce el número de bytes requeridos para almacenar un objeto del tipo de su operando. El operando es una expresión que no es evaluada o el nombre de tipo entre paréntesis.

Cuando sizeof se aplica a char, sizeof(char), el resultado es 1, , cuando se aplica aun arreglo el resultado es el número de total de bytes en el arreglo. El resultado es un entero constante sin signo. Se dice entonces que el operador sizeof permite conocer el tamaño en bytes de cualquier tipo de datos.

## Desde Un Punto De Vista Operativo

### - Para La Entrada Y Salida De Registros De Información

La función fwrite escribe un bloque de información (un número específico de bytes) a un archivo.

Por ejemplo:

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

```
struct Njaulas
```

```
{
    int Numjaulas;
    char EspeAnimal[10];
    char NomAnimal[10];
    fecha FecNacimiento;
    double peso;
    double KilogComida;
    int FrecLimpieza;
    char EstadoSalud;
    short Desendencia;
    short peligroso;
};
```

```
int main ()
```

```
{
    struct Njaulas jaula;
```

```
    FILE *archi;
```

```
    if ((archi = fopen("c:\\JAULAS.DAT", "wb"))== NULL)           // Archivos binarios
    {
        printf("imposible crear el archivo\n");
        return 1;
    }
```

```
        scanf("%d",&jaula.Numjaulas);
        gets(jaula.EspeAnimal);
        gets(jaula.NomAnimal);
        scanf("%d",&jaula.FecNacimiento.da_day);
        scanf("%d",&jaula.FecNacimiento.da_mon);
        scanf("%d",&jaula.FecNacimiento.da_year);
        scanf("%lf",&jaula.peso);
        scanf("%lf",&jaula.KilogComida);
        scanf("%d",&jaula.FrecLimpieza);
        scanf("%c",&jaula.EstadoSalud);
        scanf("%d",&jaula.Desendencia);
        scanf("%d",&jaula.peligroso);
```

```
        fwrite(&jaula, sizeof(jaula), 1, archi);
```

La función `fread` lee un número especificado de bytes en un archivo y los copia en la dirección de memoria indicada.

Por ejemplo: `fread(&jaula, sizeof(jaula),1,archi);`

---

### **Bibliografía**

- Deitel H. M. y P. J. Deitel 2000 *Como programar en C/C++*
- Gottfried B. 1997 *Programación en C*
- Kernighan B. W. y D. M. Ritchie 2001 *El lenguaje de programación C*
- Pappas Ch. y W. Murray 1995 *Manual de Borland C++*