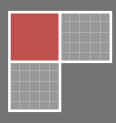
2013

Archivos de datos en C

Ficheros

Departamento de Computación



ARCHIVOS DE DATOS EN C

⇒ INTRODUCCIÓN:

En la actualidad es común procesar volúmenes de información tan grandes que es prácticamente imposible almacenar los datos en la memoria interna rápida - memoria principal- de la computadora. Estos datos se guardan generalmente en dispositivos de almacenamiento secundario como cintas y discos, en forma de archivos de datos, los cuales nos permiten almacenar la información de manera permanente y acceder a ella o modificarla cada vez que sea necesario. Los datos que hemos tratado hasta el momento han residido en la memoria principal. Sin embargo, las grandes cantidades de datos se almacenan normalmente en un dispositivo de memoria secundaria. Estas colecciones de datos se conocen como archivos (antiguamente ficheros).

Un archivo es un conjunto de datos estructurados en una colección de entidades elementales o básicas denominadas registros que son de igual tipo y constan a su vez de diferentes entidades de nivel más bajos denominadas campos.

Hay dos tipos de archivos, archivos de texto y archivos binarios.

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, fuentes de programas, base de datos simples, etc. 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.

Un archivo binario es una secuencia de bytes que tienen una correspondencia uno a uno con un dispositivo externo. Así que no tendrá lugar ninguna traducción de caracteres. Además, el número de bytes escritos (leídos) será el mismo que los encontrados en el dispositivo externo. Ejemplos de estos archivos son Fotografías, imágenes, texto con formatos, archivos ejecutables (aplicaciones), etc.

El lenguaje C ofrece un amplio conjunto de funciones de librerías para crear y utilizar archivos de datos. Lo primero que debemos hacer con los archivos de datos orientados a flujo es crear un área de buffer o de almacenamiento. Un área de buffer es un lugar de memoria que se usa para transferir datos desde la memoria a los dispositivos de almacenamiento secundario y viceversa.

Se puede conseguir la entrada y la salida de datos a un archivo a través del uso de la biblioteca de funciones; C no tiene palabras claves que realicen las operaciones de E/S. La siguiente tabla da un breve resumen de las funciones que se pueden utilizar. Se debe incluir la librería STDIO.H. Observe que la mayoría de las funciones comienzan con la letra “F”, esto es un vestigio del estándar C de Unix.

Nombre	Función
fopen()	Abre un archivo.
fclose()	Cierra un archivo.
fgets()	Lee una cadena de un archivo.
fputs()	Escribe una cadena en un archivo
fseek()	Busca un byte especifico de un archivo.
fprintf()	Escribe una salida con formato en el archivo.
fscanf()	Lee una entrada con formato desde el 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.
fflush()	Vacía un archivo.

⇒ EL PUNTERO A UN ARCHIVO:

El puntero a un archivo es el hilo común que unifica el sistema de E/S con buffer. Un puntero a un archivo es un puntero a una información que define varias cosas sobre él, incluyendo el nombre, el estado y la posición actual del archivo. En esencia identifica un archivo específico y utiliza la secuencia asociada para dirigir el funcionamiento de las funciones de E/S con buffer. Un puntero a un archivo es una variable de tipo puntero al tipo FILE que se define en STDIO.H. Un programa necesita utilizar punteros a archivos para leer o escribir en los mismos. Para obtener una variable de este tipo se utiliza una secuencia como esta: **FILE *F;**

⇒ APERTURA DE UN ARCHIVO:

La función **fopen()** abre una secuencia para que pueda ser utilizada y la asocia a un archivo.

Su prototipo es: **FILE *fopen(const char nombre_archivo, cost char modo);**

Donde:
nombre_archivo es un puntero a una cadena de caracteres que representan un nombre valido del archivo y puede incluir una especificación del directorio.
La **cadena a la que apunta modo** determina como se abre el archivo. La siguiente tabla muestra los valores permitidos para modo.

Modo	Significado
r	Abre un archivo de texto para lectura.
w	Crea un archivo de texto para escritura.
a	Abre un archivo de texto para añadir.
rb	Abre un archivo binario para lectura.
wb	Crea un archivo binario para escritura.
ab	Abre un archivo binario para añadir.
r+	Abre un archivo de texto para lectura / escritura.
w+	Crea un archivo de texto para lectura / escritura.
a+	Añade o crea un archivo de texto para lectura / escritura.
r+b	Abre un archivo binario para lectura / escritura.
w+b	Crea un archivo binario para lectura / escritura.
a+b	Añade o crea un archivo binario para lectura / escritura.

La función fopen() devuelve un puntero a archivo. Un programa nunca debe alterar el valor de ese puntero.

Si se produce un error cuando se está intentando abrir un archivo, fopen() devuelve un **puntero nulo**. Se puede abrir un archivo bien en modo texto o binario. En la mayoría de las implementaciones, en modo texto, la secuencias de retorno de carro / salto de línea se convierten a caracteres de salto de línea en lectura. En la escritura, ocurre lo contrario: los caracteres de salto de línea se convierten en salto de línea. Estas conversiones no ocurren en archivos binarios.

En lenguaje C un archivo básicamente se abre y cierra de la siguiente manera:

```
#include<stdio.h>
void main()
{
        FILE *puntero_fichero;
        puntero_fichero = fopen("nombre_archivo", "modo_apertura");
        if(puntero_fichero != NULL)
        {
                //Proceso
                fclose(puntero_fichero);
        }
        else
                printf("No se puede abrir el fichero");
}
```

Si se usa fopen() para abrir un archivo para escritura, entonces cualquier archivo existente con el mismo nombre se borrará y se crea uno nuevo. Si no existe un archivo con el mismo nombre, entonces se creará. Si se quiere añadir al final del archivo entonces debe usar el modo a. Si se usa a y no existe el archivo, se devolverá un error. La apertura de un archivo para las operaciones de lectura requiere que exista el archivo. Si no existe, fopen() devolverá un error. Finalmente, sus se abre un archivo para las operaciones de leer / escribir, la computadora no lo borrará si existe; sin embargo, si no existe, la computadora lo creará.

⇒ CIERRE DE UN ARCHIVO:

La función **fclose()** cierra una secuencia que fue abierta mediante una llamada a fopen().

Escribe toda la información que todavía se encuentre en el buffer en el disco y realiza un cierre formal del archivo a nivel del sistema operativo. Un error en el cierre de una secuencia puede generar todo tipo de problemas, incluyendo la pérdida de datos, destrucción de archivos y posibles errores intermitentes en el programa.

El prototipo de esta función es: **int fclose(FILE *F);**

Donde F es el puntero al archivo devuelto por la llamada a fopen(). Si se devuelve un valor cero significa que la operación de cierre ha tenido éxito. Generalmente, esta función solo falla cuando un disco se ha retirado antes de tiempo o cuando no queda espacio libre en el mismo.

Ejemplo #1: Programa que permite leer desde el archivo hasta que se encuentre el carácter de EOF. Usa la funcion getc para leer un carácter desde el archivo que está abierto. La funcion getc(buff) lee desde el archivo de texto y retorna el carácter a leerse. La función putchar se usa para mostrar el carácter leído.

```
#include<stdio.h>
void main()
{
        FILE *buff;
        char ch;

        /* Abrir el archivo para lectura*/
          buff= fopen("origen.txt","r");
          if(buff==NULL)
                printf("Error al abrir archivo \n");
           else
           {
                 /*leer un carácter a la vez hasta alcanzar EOF */
                  ch= getc (buff);
                  while(ch != EOF)
                  {
                            putchar(ch);
                            ch= getc(buff);
                   }
                /*cerrar archivo*/
                fclose(buff);
          }
}
```

⇒ **FUNCIONES PARA INTRODUCIR U OBTENER DATOS DE UN ARCHIVO:**

Para almacenar datos en un fichero es necesario realizar una operación de escritura, de igual forma que para obtener datos hay que efectuar una operación de lectura. En C existen muchas y variadas operaciones para leer y escribir en un fichero; entre ellas tenemos: **fread -fwrite, fgetc -fputc, fgets -fputs, fscanf -fprintf.**

Es aconsejable utilizarlas por parejas; es decir, si se escribe con fwrite se debe leer con fread.

⇒ **Lectura y escritura de caracteres (fgetc – fputc):**

fgetc: Lee un carácter del fichero. Cuando se llega al final del fichero devuelve EOF.

carácter_leido = fgetc (fichero);

El prototipo correspondiente de fgetc es: **char fgetc(FILE *archivo);**

Ejemplo #2: Programa que escribe carácter a carácter el contenido de un fichero.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        FILE *archivo;
        char caracter;
        archivo = fopen("origen.txt","r");
        if (archivo == NULL)
                printf("\nError de apertura del archivo. \n\n");
        else
        {
                printf("\nEl contenido del archivo origen es: \n\n");
                while (feof(archivo) == 0)
                {
                        caracter = fgetc(archivo);
                        printf("%c",caracter);
                }
        }
        fclose(archivo);
        return 0;
}
```

⇒ **Escritura de caracteres: fputc (car, fichero)**

Esta función escribe un carácter a la vez del archivo que está siendo señalado con el puntero *archivo. El valor de retorno es el carácter escrito, si la operación fue completada con éxito, en caso contrario será EOF.

El prototipo correspondiente de fputc es: **int fputc(int carácter, FILE *archivo);**

Ejemplo #3: Programa que escribe una cadena carácter a carácter en el fichero.

```
#include <stdio.h>
int main ( int argc, char **argv )
{
        FILE *fp;
        char caracter;
        fp = fopen ("fichero.txt", "r+" );
        printf("\nIntroduce un texto al fichero: ");
        while((caracter = getchar()) != '\n')
        {
                printf("%c", fputc(caracter, fp));
        }
        fclose ( fp );
        return 0;
}
```

⇒ Leer cadenas a o desde los ficheros: fgets()

Esta función está diseñada para leer cadenas de caracteres. Leerá hasta n-1 caracteres o hasta que lea un retorno de línea. En este último caso, el carácter de retorno de línea también es leído.

El prototipo correspondiente de fgets es:
char *fgets(char *buffer, int tamaño, FILE *archivo);

El primer parámetro buffer lo hemos llamado así porque es un puntero a un espacio de memoria del tipo char (podríamos usar un arreglo de char). El segundo parámetro es tamaño que es el límite en cantidad de caracteres a leer para la funcion fgets. Y por último el puntero del archivo por supuesto que es la forma en que fgets sabrá a qué archivo debe leer.

Ejemplo #4: Programa muestra la forma como se manejan las cadenas de caracteres en un archivo.

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
	FILE *archivo;
	char caracteres[100];
	archivo = fopen("prueba.txt","r");
	if (archivo == NULL)
		exit(1);
	printf("\nEl contenido del archivo de prueba es \n\n");
	while (feof(archivo) == 0)
	{
		fgets(caracteres,100,archivo);
		printf("%s",caracteres);
	}
	system("PAUSE");
	fclose(archivo);
}
```

La función fgets se comporta de la siguiente manera, leerá del archivo apuntado por archivo los caracteres que encuentre y a ponerlos en buffer hasta que lea un carácter menos que la cantidad de caracteres especificada en tamaño o hasta que encuentre el final de una línea (\n) o hasta que encuentre el final del archivo (EOF). El beneficio de esta función es que se puede obtener una línea completa a la vez.

⇒ Escribir cadenas a o desde los ficheros: fputs()

La función fputs escribe una cadena en un fichero. No se añade el carácter de retorno de línea ni el carácter nulo final. El valor de retorno es un número no negativo o EOF en caso de error. Los parámetros de entrada son la cadena a escribir y un puntero a la estructura FILE del fichero donde se realizará la escritura.

El prototipo correspondiente de fputs es: **int fputs(const char *buffer, FILE *archivo);**

Para ver su funcionamiento mostramos el siguiente ejemplo:

```
#include <stdio.h>
int main ( int argc, char **argv )
{
        FILE *fp;
        char cadena[] = "Mostrando el uso de fputs en un fichero.\n";
        fp = fopen ( "fichero.txt", "r+" );
        fputs( cadena, fp );
        fclose ( fp );
}
```

Ejemplo #6: Programa maneja las cadenas de caracteres en un archivo. Se introducen las instrucciones fgets y fputs para la lectura y escritura de cadenas de caracteres, respectivamente.

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
        FILE *archivo;
        int res;
        char caracteres[50];
        archivo = fopen("prueba.txt","w");
        if (archivo != NULL)
        {
                printf("Desea ingresar una cadena de caracteres? Si:1 No:0:");
                scanf("%d",&res);
                while(res)
                {
                        fflush(stdin);
                        puts("Cadena:");
                        gets(caracteres);
                        fputs(caracteres,archivo);
                        printf("Desea ingresar otra cadena de caracteres? Si:1 No:0:");
                        scanf("%d",&res);
                        if(res)
                                fputs("\n",archivo);
/*Se indica un salto de linea, excepto en la última cadena. Si no se hiciera esta indicación, la función puts pegaría las cadenas y tendríamos dificultades en el momento de leerlas. Por otra parte si realizáramos este salto de línea al final de la última cadena en la escritura se repetiría la última cadena.*/
                }
                fclose(archivo);
        }
        else
                printf("No se puede abrir el fichero\n");
}
```

⇒ Lectura y escritura formateada de texto fprintf() y fscanf()

Estas funciones se comportan exactamente como prinft() y scanf() discutidas anteriormente, excepto que operan sobre archivo. La única diferencia con las anteriores es la necesidad de dar como primer argumento el fichero en el que leemos o escribimos.

⇒ fscanf

La función fscanf funciona igual que scanf en cuanto a parámetros, pero la entrada se toma de un fichero en lugar del teclado. El prototipo correspondiente de fscanf es:

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

Ejemplo #7: Programa que abre un archivo "fichero.txt" en modo lectura y lee dentro de él.

```
#include <stdio.h>
int main ( int argc, char **argv )
{
        FILE *fp;
        char buffer[100];
        fp = fopen ( "fichero.txt", "r" );
        fscanf(fp,"%s",buffer);
        printf("%s",buffer);
        fclose ( fp );
        return 0;
}
```

⇒ fprintf

La función fprintf funciona igual que printf en cuanto a parámetros, pero la salida se dirige a un archivo en lugar de a la pantalla. El prototipo correspondiente de fprintf es:

int fprintf(FILE *archivo, const char *formato, argumento, ...);

Ejemplo #8: Programa que abre el "fichero.txt" en modo lectura/escritura y escribe dentro de él.

```
#include <stdio.h>
int main ( int argc, char **argv )
{
        FILE *fp;
        char buffer[100] = "Esto es un texto dentro del fichero.";
        fp = fopen ( "fichero.txt", "r+" );
        fprintf(fp, buffer);
        fprintf(fp, "%s", "\nEsto es otro texto dentro del fichero.");
        fclose ( fp );
        return 0;
}
```

Ejemplo #9: Programa en C que lee de un archivo el número de alumnos, número de carnet y las cinco calificaciones de cada uno de ellos e imprime en pantalla el número de carnet y el promedio de cada alumno.

```
#include <stdio.h>
#include <stdlib.h>
void main()
{
        FILE *pf;
        int i,j,n;
        float calif,prom=0,sum=0.0;
        char carnet[11];
        if((pf = fopen("matricula.txt","r")) != NULL)
        {
                fscanf(pf,"%d",&n);
                for(i=0;i<n;i++)
                {
                        fscanf(pf,"%s",carnet);
                        printf("%s\t",carnet);
                        sum=0.0;
                        for(j=0;j<3;j++)
                        {
                                fscanf(pf,"%f",&calif);
                                sum+=calif;
                        }
                        prom=sum/3;
                        printf("\t%.2f ",prom);
                        printf("\n");
                }
                fclose(pf);
        }
        else
                printf("No se puede abrir el fichero\n");
}
```

⇒ **Lectura y escritura de bloques (fread – fwrite):**

fread: Esta función lee un bloque de una "stream" de datos. Efectúa la lectura de un arreglo de elementos "count", cada uno de los cuales tiene un tamaño definido por "size". Luego los guarda en el bloque de memoria especificado por "ptr". El indicador de posición de la cadena de caracteres avanza hasta leer la totalidad de bytes. Si esto es exitoso la cantidad de bytes leídos es (size*count).

El prototipo correspondiente de fread es:
size_t fread (void * ptr, size_t size, size_t count, FILE * stream);

PARAMETROS:
ptr : Puntero a un bloque de memoria con un tamaño mínimo de (size*count) bytes.
size : Tamaño en bytes de cada elemento (de los que voy a leer).
count : Número de elementos, los cuales tienen un tamaño "size".

stream: Puntero a objetos FILE, que especifica la cadena de entrada.

fwrite: Esta función está pensada para trabajar con registros de longitud constante y forma pareja con fread. Es capaz de escribir hacia un fichero uno o varios registros de la misma longitud almacenados a partir de una dirección de memoria determinada. El valor de retorno es el número de registros escritos, no el número de bytes. Los parámetros son: un puntero a la zona de memoria de donde se obtendrán los datos a escribir, el tamaño de cada registro, el número de registros a escribir y un puntero a la estructura FILE del fichero al que se hará la escritura.

El prototipo correspondiente de fwrite es:
size_t fwrite(void *puntero, size_t tamano, size_t cantidad, FILE *archivo);

Un ejemplo concreto del uso de fwrite con su contraparte fread y usando funciones es:

```
#include <stdio.h>
void menu();
void CrearFichero(FILE *Fichero);
void InsertarDatos(FILE *Fichero);
void VerDatos(FILE *Fichero);

struct sRegistro
{
	char Nombre[25];
	int Edad;
	float Sueldo;
} registro;

int main()
{
	int opcion;
	int exit = 0;
	FILE *fichero;
	while (!exit)
	{
		menu();
		printf("\nOpcion: ");
		scanf("%d", &opcion);
		switch(opcion)
		{
			case 1:
				CrearFichero(fichero);
				break;
			case 2:
				InsertarDatos(fichero);
				break;
			case 3:
				VerDatos(fichero);
				break;
```

```
                        case 4:
                                exit = 1;
                                break;
                        default:
                                printf("\nopcion no valida");
                }
        }
        return 0;
}

void menu()
{
        printf("\nMenu:");
        printf("\n\t1. Crear fichero");
        printf("\n\t2. Insertar datos");
        printf("\n\t3. Ver datos");
        printf("\n\t4. Salir");
}

void CrearFichero(FILE *Fichero)
{
        Fichero = fopen("fichero", "r");
        if(!Fichero)
        {
                Fichero = fopen("bloques.txt", "w");
                printf("\nArchivo creado!");
        }
        else
        {
                printf("\nEl fichero ya existe!");
        }
        fclose (Fichero);
        return;
}

void InsertarDatos(FILE *Fichero)
{
        Fichero = fopen("fichero", "a+");
        if(Fichero == NULL)
        {
        printf("\nFichero no existe! \nPor favor creelo");
        return;
        }
        printf("\nDigita el nombre: ");
        scanf("%s", &registro.Nombre);
        printf("\nDigita la edad: ");
        scanf("%d", &registro.Edad);
        printf("\nDigita el sueldo: ");
        scanf("%f", &registro.Sueldo);
```

```
        fwrite(&registro, sizeof(struct sRegistro), 1, Fichero);
        fclose(Fichero);
        return;
}

void VerDatos(FILE *Fichero)
{
        int numero = 1;
        Fichero = fopen("fichero", "r");
        if(Fichero == NULL)
        {
                printf("\nFichero no existe! \nPor favor creelo");
                return;
        }
        fread(&registro, sizeof(struct sRegistro), 1, Fichero);
        printf("\nNumero \tNombre \tEdad \tSueldo");
        while(!feof(Fichero))
        {
                printf("\n%d \t%s \t%d \t%.2f", numero, registro.Nombre,
                registro.Edad, registro.Sueldo);
                fread(&registro, sizeof(struct sRegistro), 1, Fichero);
                numero++;
        }
        fclose(Fichero);
        return;
}
```

⇒ **EJERCICIOS PROPUESTOS:**

1. Mostrar el contenido de un fichero en mayúsculas usando fgetc.
2. Leer el contenido de un fichero y sustituír las ocurrencias de la letra 'a' por la letra 'x'.
3. Copiar un fichero en otro utilizando fgets y fputs().
4. Crear un programa en C que guarde en un fichero los meses del año guardados en un array meses y luego leer el fichero y mostrarlos.
5. Crear un programa que guarde en un fichero "estudiantes.dat" los datos siguientes de estudiantes:

```
struct estudiante
{
        int edad;
        char nombre[20];
        float notas[3];
};
```

Este programa contendrá un menú con lo siguiente:
a) Ingresar estudiante
b) Mostrar un estudiante
c) Mostrar estudiantes

⇒ BIBLIOGRAFÍA BÁSICA:

- Ceballos, Francisco Javier: **C/C++ Curso de Programación**, 2da Edición. Editorial RA-MA, 2002.
- Joyanes Aguilar, Luis; Zahonero Martínez Ignacio: **Programación en C.** McGraw Hill, 2001.
- Gottfried, Byron S: **Programación en C**. McGraw Hill, 1991.