Tema 6 . Gestión de ficheros en C

Programación en Lenguajes Estructurados

Tema 6 . Gestión de ficheros en C

6.1. Introducción a los ficheros

En la mayor parte de los programas se quiere almacenar información que no desaparezca al finalizar la ejecución de los mismos. Esto es lo que diferencia el almacenamiento de datos en ficheros al almacenamiento de datos en estructuras, vectores, etc...

Antiguamente el C trataba bajo Unix los ficheros como una sucesión de octetos (grupos de 8 bits) pudiendo accederse a cada uno de forma individual. En posteriores generalizaciones del C en sistemas operativos diferentes se modificó la forma de tratar y usar los ficheros por parte del C, creándose una serie de librerías para facilitar su manipulación.

Este es el caso de la librería **stdio.h** (acrónimo de standard input/output), que define estructuras de tipo fichero y un conjunto de funciones que nos facilitarán el trabajo con los ficheros. Es por ello que todas nuestras implementaciones en C comenzarán con la inclusión de la cabecera

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

A continuación declararemos una variable de tipo FILE como mostramos a continuación:

```
FILE *fichero;
```

Mencionar que cuando un programa en C comienza a ejecutarse se abren 3 flujos de memorias o buffers para la entrada estándar, salida estándar y salida de errores. Estas variables son **stdin**, **stdout** y **stderr**.

Por entrada estándar entendemos el dispositivo físico por donde se introducen los datos al programa, normalmente el teclado. Por defecto, la salida estándar es el monitor aunque se puedan cambiar estos aspectos.

6.2. Abrir y cerrar ficheros

Las primeras opciones que vamos a ver son la apertura y clausura o cierre de un fichero: **fopen** y **fclose**. Los prototipos de estas funciones son:

*file *fopen(char *nombre_fichero, char *modo);*

*int fclose(FILE *fich);*

PSEUDOCÓDIGO

Declaración de variables de tipo fichero: Fichero F1

Abrir (nombre_fichero, modo)
Cerrar(nombre_fichero)

Una vez abierto el fichero con *fopen*, se dispondrá de un puntero a una estructura FILE desde el que puede acceder al contenido del mismo. Pero realmente no se apunta al fichero sino a un buffer o almacén temporal que tiene una copia del mismo. Los datos no se volcarán en el fichero hasta que se realice el *fclose*.

Se pueden producir errores como que el disco esté lleno, el nombre del fichero sea ilegal, el acceso al fichero esté restringido, que se haya sacado el disquete o exista algún otro problema con los dispositivos físicos.

Si no se consigue abrir el fichero, *fopen* devolverá *NULL*. Por otra parte, *fclose* devolverá un 0 cuando el fichero se cierre sin problemas. Hay que verificar siempre esas circustancias.

La función fopen recibe como argumentos una cadena de caracteres correspondiente al nombre del fichero y otro parámetro será la forma de interactuar con el fichero. Estos valores que nos indica la posibilidad de cómo interactuar son:

Tipo	Descripción
r	Abre el fichero para la lectura. El fichero debe existir previamente; en otro caso se produce un error en la apertura. Proviene de READ.
w	Abre el fichero para la escritura. Si no existe, se crea y si existe se destruye el contenido del anterior. Proviene de WRITE.
a	Abre el fichero para escritura desde el final del mismo. Es decir, añade. Si el fichero no existe, se crea uno nuevo. Proviene de APPEND.
r+	Abre el fichero, tanto para lectura como para escritura, por el comienzo. Debe existir el fichero, si no se produce un error.
w+	Abre el fichero por el comienzo, tanto para la lectura como para escritura. Si ya existe, el contenido previo se borra. Si no existe, se crea.
a+	Abre el fichero para lectura y escritura por el final. Si no existe previamente, se crea un fichero nuevo.

En cambio, la función *fclose* únicamente recibe como argumento el puntero a la estructura FILE asociada al fichero que queremos cerrar. A continuación tenemos un ejemplo de programa que abre y cierra un fichero verificando las pertinentes características.

```
#include <stdio.h>

main()
{
        FILE  *fichero;
        fichero = fopen("articulo.dat","r");
        if (fichero == NULL)
                printf("No se pudo abrir el fichero");
        else
                printf("Se ha abierto el fichero");
                ...

/* Aquí se operaría con el fichero. */

        If (fclose(fichero) != 0)
                printf("No se puede cerrar el fichero");
        else
                printf("Se ha cerrado el fichero");
}
```

```
Inicio
Fichero FILE
fichero = Abrir("articulo.dat","r")
Si (fichero = NULL)
        Escribir "No se pudo abrir el fichero"
Sino
        Escribir "Se ha abierto el fichero"
                ...

/* Aquí se operaría con el fichero. */

Si (Cerrar(fichero) <> 0)
        Escribir "No se puede cerrar el fichero"
Sino
        Escribir "Se ha cerrado el fichero"
```

6.3. Las funciones para trabajar con caracteres individuales

Una vez abierto el fichero, se puede leer y escribir en él. Las funciones adecuadas más sencillas para ello son *getc* y *putc*, funciones para leer y escribir un simple carácter. Los prototipos son:

*int getc(FILE *fich);*

*int putc(int c, FILE *fich);*

PSEUDOCÓDIGO

Leer_caracter (nombre_fichero)
Escribir_caracter(carácter, nombre_fichero)

Las dos funciones reciben el puntero a la estructura FILE. *putc* recibe el carácter a escribir en el fichero como argumento y *getc* devuelve el carácter leído.. Su funcionamiento es similar a *getchar* y *putchar*.

A continuación vamos a ver el ejemplo de copiar el contenido de un fichero, llamado "quijote.txt" en otro llamado "copiaquijote.txt".

```
#include <stdio.h>

main()
{
        FILE  *fich1, *fich2;
        char c;
        fich1 = fopen("quijote.txt","r");
        fich2 = fopen("copiaquijote.txt","w");
        if ((fich1 == NULL) || (fich2 ==NULL))
                printf  ("Error al abrir ficheros");
        else
        {
                while ((c=getc(fich1)) != EOF)
                        putc (c, fich2);
        if (fclose(fich1)!=0)
                printf ("Error al cerrar el fichero quijote");
        if (fclose(fich2)!=0)
                printf ("Error al cerrar el fichero copiaquijote");
        }
}
```

Conviértelo en pseudocódigo.

Si el fichero quijote.txt se hubiera abierto indicando la operación correspondiente a añadir (a+), se podría haber leído el contenido del fichero pero no podríamos haber leído desde el principio como tenemos que hacer para poder recorrer el fichero con el fin de copiarlo, pues con la opción de añadir nos situamos al final del mismo.

Ahora bien, para solventar este problema tenemos la función:

*void rewind (FILE *fich);*

PSEUDOCÓDIGO

Rebobinar_Fichero(nombre_fichero)

Con el uso de esta función, la copia de ficheros sería:

```
#include <stdio.h>

main()
{
        FILE  *fich1, *fich2;
        char c;
        fich1 = fopen("quijote.txt","a+");
        fich2 = fopen("copiaquijote.txt","w");
        if ((fich1 == NULL) || (fich2 ==NULL))
                printf  ("Error al abrir ficheros");
        else
        {
                Rewind(fich1);
                while ((c=getc(fich1)) != EOF)
                        putc (c, fich2);
        if (fclose(fich1)!=0)
                printf ("Error al cerrar el fichero quijote");
        if (fclose(fich2)!=0)
                printf ("Error al cerrar el fichero copiaquijote");
        }
}
```

A su vez tenemos una función muy parecida que es *ungetc* que deshace la última operación realizada con un *getc*. La consecuencia es que la próxima invocación de *getc* volverá a leer el carácter devuelto por *ungetc*. Su prototipo es:

*int ungetc (int c, FILE *fich);*

PSEUDOCÓDIGO

DesLeer_caracter(nombre_fichero)

Para ilustrar la utilidad, vamos a seguir con el ejemplo de copia del fichero quijote, pero vamos a copiarlo sin espacios. Para ello utilizaremos la función ungetc.

```
#include <stdio.h>

int salta_espacios(FILE *fichero);
main()
{
        FILE  *fich1, *fich2;
        char c;
        fich1 = fopen("quijote.txt","r");
        fich2 = fopen("copiaquijote.txt","w");
        if ((fich1 == NULL) || (fich2 ==NULL))
                printf  ("Error al abrir ficheros");
        else
        {
                while ((c=getc(fich1)) != EOF)
                {
                        putc (c, fich2);
                        salta_espacios(fich1);
                }
        if (fclose(fich1)!=0)
                printf ("Error al cerrar el fichero quijote");
        if (fclose(fich2)!=0)
                printf ("Error al cerrar el fichero copiaquijote");
        }
}

int salta_espacios(FILE *fichero)
{
        char letra=' ';
        while (letra==' ')
                letra = getc(fichero);
        ungetc (letra,fichero);
        return 0;
}
```

6.4. Las funciones para trabajar con cadenas de caracteres

Se pueden distinguir 2 grupos de funciones dedicadas a leer y escribir cadenas de caracteres: *fprintf* y *fscanf* por un lado y por el otro *fgets* y *fputs*. Los prototipos de estas funciones las mostramos a continuación:

*int fscanf(FILE *fich, char *formato, <lista_de_elementos>);*

*int fprintf(FILE *fich, char *formato, <lista_de_elementos>);*

Veamos a continuación el ejemplo de copia del fichero quijote.txt en copiaquijote.txt palabra a palabra.

```
#include <stdio.h>

main()
{
        char palabras[20];
        FILE  *fich1, *fich2;
        char c;
        fich1 = fopen("quijote.txt","r");
        fich2 = fopen("copiaquijote.txt","w");
        if ((fich1 == NULL) || (fich2 ==NULL))
                printf  ("Error al abrir ficheros");
        else
        {
                while ((fscanf(fich1, "%s", palabras)==1)
                        fprintf(fich2, "%s", palabras);
        if (fclose(fich1)!=0)
                printf ("Error al cerrar el fichero quijote");
        if (fclose(fich2)!=0)
                printf ("Error al cerrar el fichero copiaquijote");
        }
}
```

Las otras funciones que nos permiten leer y escribir cadenas de caracteres de ficheros son *fgets* y *fputs*. Sus prototipos son:

*char *fgets(char *cadena, int numero, FILE *fich);*

*int fputs(char *cadena, FILE *fich);*

Son funciones equivalentes al *gets* y al *puts* que leían los datos desde teclado.

La función *fgets* tiene 3 argumentos a saber:

- El primero es la cadena de caracteres donde va a almacenar la entrada que va a leer.
- El segundo parámetro es el tamaño máximo de caracteres que va a leer.
- Y por último, el fichero de donde leer.

Por el contrario, *fputs* solo admite 2 parámetros a saber:

- El primero es la cadena de caracteres que se va a escribir en el fichero.
- Y en segundo lugar, el fichero donde escribir.

Veamos un ejemplo:

```
#include <stdio.h>

main()
{
        Char palabras[20];
        FILE  *fich1, *fich2;
        char c;
        fich1 = fopen("quijote.txt","r");
        fich2 = fopen("copiaquijote.txt","w");
        if ((fich1 == NULL) || (fich2 ==NULL))
                printf  ("Error al abrir ficheros");
        else
        {
                while (((fgets(palabras,20,fich1) !=NULL) &&
                        (palabras[0]!='\n'))
                        fputs(palabras, fich2);
        if (fclose(fich1)!=0)
                printf ("Error al cerrar el fichero quijote");
        if (fclose(fich2)!=0)
                printf ("Error al cerrar el fichero copiaquijote");
        }
}
```

Ejercicios

1. Realiza un algoritmo en pseudocódigo y en C que abra un fichero y grabe en él 2 palabras que han sido pedidas por teclado.

2. Realiza un algoritmo en pseudocódigo y en C que lea el fichero anterior y muestre por pantalla las 2 palabras almacenadas en él.

6.5. Las funciones para trabajar con contenidos binarios

Las funciones que acabamos de ver nos permiten trabajar con ficheros cuyo contenido es un texto, es decir, una secuencia de caracteres. Pero se debe permitir trabajar con el contenido binario de un fichero. A este tipo de ficheros se puede acceder en el orden en que se desee a diferencia de los ficheros de texto cuyo acceso es secuencial.

La función que utilizamos para acceder de forma aleatoria al contenido de ficheros binarios es *fseek*. Su prototipo es el siguiente:

*int fseek (FILE *fich, long num_bytes, int origen);*

Mediante esta función podemos aleatoriamente a un fichero, posicionándonos directamente en un determinado byte o posición. Los argumentos utilizados son:

- El primero es el fichero al que queremos acceder.
- El segundo parámetro es el desplazamiento, que se mide en bytes, que queremos realizar, positivo si es hacia adelante o negativo si es hacia detrás.
- El tercer argumento es la posición a partir de donde comenzamos el desplazamiento. Puede tener 3 posibilidades:
    - SEEK_SET que es el comienzo del fichero.
    - SEEK_CUR que es la posición actual.
    - SEEK_END que es el final del fichero.

fseek devolverá 0 en condiciones normales y -1 con algún error, por lo que hay que comprobar su funcionamiento. Una función muy relacionada es *ftell*.

Esta devuelve un valor entero de tipo long que se corresponde con la posición actual que nos encontramos dentro del fichero. Su prototipo es:

*long int ftell (FILE *fich);*

Veamos a continuación un ejemplo que muestra un fichero al revés.

```
#include <stdio.h>

main()
{
        char ch;
        FILE  *fich;
        long i,tam;
        fich = fopen("quijote.txt","r");
        if (fich == NULL)
                printf("Error al abrir ficheros");
        else
        {
                fseek(fich, 0L, SEEK_END);
                tam = ftell(fich);
                i=1L;
                while (i<=tam)
                {
                        fseek(fich, -i, SEEK_END);
                        ch=getc(fich);
                        if ((ch!=CNTLZ) && (ch!='\r'))
                                putchar(ch);
                        i++;
                }
                if  (fclose(fich)!=0)
                        printf("Error al cerrar fichero quijote");
        }
}
```

6.6. Funciones informativas de errores sobre ficheros

Existe un conjunto de operaciones con ficheros desarrolladas en funciones para tal fin que nos facilitan la labor. La primera es *feof*, que nos informa si hemos alcanzado el fin del fichero y la segunda es *ferror*, que informa si se ha producido algún error en el fichero.

Ambas reciben como único parámetro el puntero a FILE asociado al fichero. Sus prototipos son:

*int feof (FILE *fich);*

*int ferror (FILE *fich);*

feof devuelve un valor distinto de cero si se ha encontrado el final del fichero. Y *ferror* devuelve un valor distinto de cero si ha ocurrido un error.

Ejercicios

3. Escribe un algoritmo en pseudocódigo y en C que cuente las palabras contenidas en un fichero.

4. Escribe un algoritmo en pseudocódigo y en C que lea los datos personales de alguien, los almacena en una estructura la cual posteriormente grabará en un fichero y la leerá de él.