Archivos secuenciales en C

Introducción

Al finalizar un programa, los datos almacenados en variables se pierden. Si se requiere que esos datos se conserven en forma permanente es necesario guardarlos en archivos. Los archivos se grabarán en algún dispositivo de almacenamiento.

En C, es posible crear, actualizar y procesar archivos de datos. El lenguaje C ve a los archivos como un flujo secuencial de bytes. Si el contenido del archivo es un conjunto de valores ASCII imprimibles, separados por saltos de línea (ASCII 13) entonces tenemos un **archivo de texto**.

Si, en cambio, el archivo contiene bytes con cualquier valor del intervalo [0, 255] entonces es un **archivo binario** y su estructura interna depende de cada archivo en particular. C provee funciones que facilitan el tratamiento de ambos tipos de archivos. En este módulo, solo vamos a trabajar con archivos secuenciales de texto.

Estructura File

Antes de que un archivo pueda ser leído o modificado, debe abrirse. Para ello existe la función **fopen.**

```
FILE   *fopen(char *nombre, char *modo);
```

Dicha función toma un nombre real de archivo (una cadena de texto con la ubicación y el nombre del archivo), efectúa la negociación correspondiente con el sistema operativo y retorna un puntero a una estructura que contiene la información necesaria para manejar dicho archivo. Permite almacenar información de la conexión entre el archivo físico y el programa, por lo que podemos denominarlo el archivo lógico. Entre otras cosas, contiene la ubicación del buffer, la posición del carácter actual en el buffer, un indicador que muestra si el archivo está siendo escrito o leído, y si se ha alcanzado el fin de archivo. La estructura depende del sistema operativo y se puede ver la estructura tipo FILE en la librería stdio.h.

Ejemplo:

```
FILE *fp;
fp = fopen(“miarchivo.txt”,”w”);
```

La variable fp es un puntero a FILE. Al efectuar fopen con el nombre de un archivo en particular, y para un determinado modo (w es para escritura), retorna un puntero a una estructura de tipo FILE que se almacena en fp. A partir de entonces, todo lo que haya que hacer sobre el archivo se hará a través de dicha variable.

Para cerrar el archivo y liberar el puntero se efectúa:

```
fclose(fp);
```

Modos de apertura de un archivo

El modo de apertura de un archivo es una cadena de texto que indica cómo se usará el archivo:

`w`	*Write* (escribir). No debe existir. Si existe, se elimina.
`r`	*Read* (leer). Debe existir, si no da error.
`a`	append (agregar). Escritura al final. Si no existe, se crea vacío.
`w+`	Lectura y escritura. No debe existir. Si existe, se elimina.
`r+`	Lectura y escritura. Si no existe, se crea vacío.
`a+`	Lectura y escritura al final. Si no existe, se crea vacío.

Si hay un error, fopen retorna un puntero nulo (NULL) así que es conveniente siempre validar si el puntero obtenido no es NULL:

```
if((fp = fopen(“miarchivo.txt”, w))== NULL)
    printf(”Error al abrir el archivo”);
```

Lectura y grabación de datos en un archivo

La biblioteca estándar de C provee funciones que facilitan la operación sobre archivos de texto. Así, para la lectura de caracteres de un archivo tenemos la función fscanf:

Esta función se usa para obtener valores con un determinado formato. Dichos valores se obtienen del archivo y se almacenan en los argumentos variables.

fscanf(FILE *fp, char *format, ...);

La función `fprintf` se usa para escribir en un archivo en un determinado formato.

fprintf(FILE *fp, char *format, ...);

Control de errores

En el manejo de archivos, es muy importante controlar los errores que podrían suceder. Por ejemplo, querer efectuar una lectura más allá del fin de archivo, o querer escribir en un archivo abierto solo para lectura dan errores que deben controlarse.

Para ello, contamos con la siguiente función:

int feof (FILE * stream);

La función feof indica si se ha llegado al final del archivo. Retorna un valor distinto de cero si se alcanzó fin de archivo y cero en caso contrario.

Código para crear y leer un archivo en C

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

int main ()
{
int legajo;
char nom[15];
float sueldo;
FILE *cfptr;
if ((cfptr=fopen("e:\clientes.dat","w"))==NULL)
    printf("\n No se puede Abrir!!");
    else
    {
    printf("Ingrese Legajo (0 para terminar):");
    scanf("%d",&legajo);
    while(legajo != 0)
    {
        printf("Ingrese Nombre: \n");
        scanf("%s",nom);
        printf("Ingrese Sueldo: \n");
        scanf("%f",&sueldo);
        fprintf(cfptr,"%d  %s  %.2f\n",legajo,nom,sueldo);
        printf("Ingrese Legajo (ingrese 0 para Terminar):");
        scanf("%d",&legajo);

    }
        }
fclose(cfptr);
printf("\nLos Datos Ingresados son: \n\n");
if ((cfptr=fopen("e:\clientes.dat","r"))==NULL)
    printf("\n No Se puede Abrir!!!.");
    else
    {
    while(!feof(cfptr))
    {
        fscanf(cfptr,"%d  %s  %f\n",&legajo, nom, &sueldo);
        printf("Legajo: %d   Nombre: %s   Sueldo: %.2f\n",legajo,nom,sueldo);
    }
    }
fclose(cfptr);
}
```

El puntero cfptr recibe el puntero que devuelve fopen

fopen tiene dos parámetros: la ubicación y el nombre del archivo, y para qué se va a abrir el archivo.

fprintf requiere de tres parámetros: el puntero, el formato de las variables que va a grabar en el buffer y dichas variables.

fscanf requiere de tres parámetros:

el puntero, el formato de las variables que va a leer del archivo y dichas variables (no olvidar el &).

Cada vez que se abre el archivo, se debe cerrar con fclose.