Práctica No. 8

Manejo de archivos: Acceso secuencial

Objetivo: Crear programas de software donde se almacene la información en memoria secundaria, seleccionando de manera responsable para cada escenario el tipo de archivo que sea adecuado y así garantizar el uso eficiente de los recursos y la integridad de los datos.

Material:
- Computadora Personal (PC)
- Programa Editor de texto (ASCII), compilador GCC

Equipo:
- Computadora Personal

Introducción

El almacenamiento de los datos dentro de variables y arreglos es temporal; todos los datos se pierden cuando termina el programa. Los archivos se utilizan para retener permanentemente grandes cantidades de datos. Las computadoras almacenan los archivos en dispositivos secundarios de almacenamiento, en especial en dispositivos de disco.
C ve a cada archivo simplemente como un **flujo secuencial de byte**s. Cada archivo termina con una marca de fin de archivo o con un número de byte específico almacenado dentro de una estructura de dato administrativa mantenida por el sistema. Cuando se abre un archivo, se le asocia un flujo. Cuando comienza la ejecución de un programa, se abren tres archivos asociados y sus flujos asociados (de entrada estándar **`stdin`**, de salida estándar **`stdout`**, y de error estándar **`stderr`**). Los flujos proporcionan canales de comunicación entre los archivos y los programas. Por ejemplo, el flujo estándar de entrada permite a un programa leer datos desde el teclado, y el flujo estándar de salida permite al programa desplegar los datos en la pantalla.
Al abrir un archivo se devuelve un apuntador a la estructura FILE (definida en `stdio.h`), la cual contiene información utilizada para procesar el archivo. Esta estructura incluye un descriptor de archivo, es decir, un índice dentro de un arreglo del sistema operativo llamado tabla de archivos abiertos. Cada elemento del arreglo contiene un bloque de control de archivo (FCB) que utiliza el sistema operativo para administrar un archivo en particular. La entrada estándar, la salida estándar y el error estándar se manipulan por medio de los apuntadores de archivo `stdin`, `stdout` y `stderr`.

El formato para declarar un archivo es :

```
FILE *var_archivo;
```

FILE es una estructura de datos definida en stdio.h que contiene información acerca del archivo o flujo de datos al cual se desea acceder. Alguna de la información que contiene la estructura `FILE` es la dirección del buffer que utiliza, modo de apertura del archivo, último carácter leído del buffer, entre otra información usualmente no requerida por el usuario.

Las funciones de manipulación de archivos en stdio.h requiere como parámetro un apuntador a una estructura del tipo `File`, tales como, fopen, feof, fclose, fprintf, fscanf, etc.

APERTURA DE UN ARCHIVO

La apertura de un archivo es la creación física y lógica del mismo en el programa de aplicación.
Verifica si existe o no el archivo y direcciona su posición en el lugar de almacenamiento en disco. La función de apertura y su formato es :

```
var_archivo = fopen(nombre_en_disco_archivo, clave_de_apertura);
```

Algunas de las posibles claves de apertura de archivos son las siguientes :

Clave	Función
w	Crear archivo nuevo, si existe se elimina contenido
a	Agregar datos a un archivo
r	Apertura de solo lectura
w+	Crear archivo nuevo para escribir y leer
a+	Agregar datos a un archivo, si no existe es como w+
r+	Apertura para leer y escribir

Ejemplo :

```
arch = fopen("archivo.dat" , "w" );
```

Las funciones relacionadas con el manejo de archivos en el lenguaje C que se encuentran en la biblioteca stdio.h requieren como parámetro un apuntador a una estructura FILE de un flujo por abrir o ya abierto. Algunas de las funciones relacionadas con el acceso de dato secuencial se encuentra en la siguiente tabla.

Función	**Descripción**
FILE * fopen (const char * filename, const char * mode);	Abre un archivo en el modo indicado y regresa un apuntador a una estructura File que contiene la información necesaria para su manipulación.
int fclose (FILE * stream);	Cierra un flujo de datos abierto anteriormente con fopen.
int feof (FILE * stream);	Verifica el indicador de fin de línea.
int remove (const char * filename);	Elimina un archivo indicado en **filename**.
int rename (const char * oldname, const char * newname);	Cambia el nombre de un archivo.
int fgetc (FILE * stream);	Regresa el caracter que se encuentra actualmente apuntado por el indicador de posición del flujo.
char * fgets (char * str, int num, FILE * stream);	Lee un conjunto de caracteres de tamaño **num** que son guardados en un apuntador **str**.
int fscanf (FILE * stream, const char * format, ...);	Lee un conjunto de datos con un determinado formato indicado en **format** en un flujo de datos.
int fputc (int character, FILE * stream);	Escribe un caracter en el flujo de datos indicado por **stream.**
int fputs (const char * str, FILE * stream);	Escribe una cadena de caracteres en un flujo de datos indicada en el apuntador **stream**. La cadena de caracteres se escribe en el flujo de datos hasta alcanzar

	el carácter ´\0´.
int fprintf (FILE * stream, const char * format, ...);	Escribe en el flujo indicado en **stream** la información pasada en la cadena **char** y valores de las variables si se pasan a la función.

Desarrollo de la práctica

Implementar los siguientes programas en archivos independientes .c. Cada uno de los programas debe recibir como parámetro por consola el nombre del archivo(s) a manipular.

1. Implementar un programa (`more.c`) para leer un archivo de texto e imprimir su contenido en la consola. Si el archivo cuenta con más de 15 líneas de texto, se deberá realizar una pausa en la lectura hasta que el usuario presione enter y se muestran las siguientes 10 líneas si las tiene.

2. Elaborar un programa (`invertir.c`) que lea un archivo de texto e invierta las minúsculas por mayúsculas y viceversa, además, dependiendo de los parámetros por consola imprimir el contenido del archivo en la consola o en otro archivo.
   ```
   >invertir.exe nombre_de_archivo.txt
   >invertir.exe nombre_de_archivo.txt nombre_destino.txt
   ```

3. Elaborar un programa que lea de un archivo de texto con una serie de números enteros, al leer cada número se debe ir guardando en un arreglo de tamaño dinámico, una vez terminado de leer el archivo, imprimir el arreglo creado, el valor máximo, mínimo, media, mediana y moda.
   a. Hacer uso de arreglos de apuntadores a función.
   b. Las funciones deberán contener el siguiente prototipo.
      ```
      float funcion(int* array, int size);
      ```
   c. El programa debe de ejecutar de forma iterativa el arreglo de apuntadores a funciones, apoyarse en el código de ejemplo.
   d. La función para ordenar elementos de un arreglo es libre, se recomienda consultar la función qsort de `stdlib.h`.

Restricciones

1. El código fuente deberá contener comentarios que indique el objetivo del programa y descripción de instrucciones más relevantes.
2. Evitar nombres de variables y funciones que no reflejen su propósito claramente.
3. Cuando se haga uso de memoria dinámica es importante no olvidar liberar la memoria.

Conclusiones y Comentarios.