Laboratorios de computación

salas A y B

Profesor: Claudia Rodríguez Espino

Asignatura: Fundamentos de Programación

Grupo: 4

No. de Práctica: 13

Integrante: Sanchez Escamilla Hector

No. De equipo de cómputo: 35

Semestre: 2

Práctica 13: Lectura y escritura de datos

**Objetivo:**

Elaborar programas en lenguaje C que requieran el uso de archivos de texto plano en la resolución de problemas, entendiendo a los archivos como un elemento de almacenamiento secundario.

**Actividades:**

- A través de programas en C, emplear las funciones para crear, leer, escribir y sobrescribir archivos de texto plano.

- Manipular archivos empleando los diferentes tipos de acceso a ellos.

**Introducción**Un archivo es un conjunto de datos estructurados en una colección de entidades elementales o básicas denominadas registros que son del mismo tipo, pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.  
Lenguaje C permite manejar la entrada y la salida de datos desde o hacia un archivo, respectivamente, a través del uso de la biblioteca de funciones de la cabecera *stdio.h*.

**ARCHIVOS**

Un archivo es un conjunto de datos estructurados en una colección de entidades elementales o básicas denominadas registros que son del mismo tipo, pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Lenguaje C permite manejar la entrada y la salida de datos desde o hacia un archivo respectivamente, a través del uso de la biblioteca de funciones de la librería stdio.h

Un apuntador a un archivo es un hilo común que unifica el sistema de E/S con un buffer donde se transportan los datos.

Un apuntador a un archivo señala a la información que contiene y define ciertas características sobre él, incluyendo el nombre, el estado y la posición actual del archivo.

Los apuntadores a un archivo se manejan en lenguaje C como variables apuntador de tipo FILE que se define en la librería stdio.h. La sintaxis para obtener una variable apuntador de archivo es la siguiente

FILE \*F

**ABRIR ARCHIVO**

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

Su estructura es:

\*FILE fopen (char \*nombre\_archivo, char \*modo);

Donde nombre\_archivo es un apuntador una cadena de caracteres que representan un nombre válido del archivo y puede incluir una especificación del directorio. La cadena a la que apunta modo determina como se abre el archivo.

**CERRAR ARCHIVO**

La función fclose() cierra una secuencia que fue abierta mediante una llamada a fopen(). 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 perdida de datos, destrucción de archivos y posible errores intermitentes en el programa, la función es:

int fclose(FILE \*apuntArch);

Donde apuntArch es el apuntador al archivo devuelto por la llamada a fopen(). Si se devuelve un cero significa que la operación del cierre ha tenido éxito, si falla generalmente es cuando no queda espacio libre o cuando un disco se ha retirado antes de tiempo.

**FUNCIONES fgets y fputs**

Las funciones fgets() y fputs() pueden leer y escribir, respectivamente, cadenas sobre los archivos.

char \*fgets(char \*buffer, int tamaño, FILE \*apuntArch);

char \*fputs(char \*buffer, FILE \*apuntArch);

La función fputs() permite escribir una cadena en un archive especifico.

La función fgets() permite leer una cadena desde el archivo especificado. Esta función lee un renglón al a vez.

FUNCIONES fscanf y fprintf

Las funciones fscanf() y fprintf() se comportan exactamente como printf() y scanf(), excepto que operan sobre archivo. Sus estructuras son:

int fprintf(FILE \*apuntArch, char \*format,…);

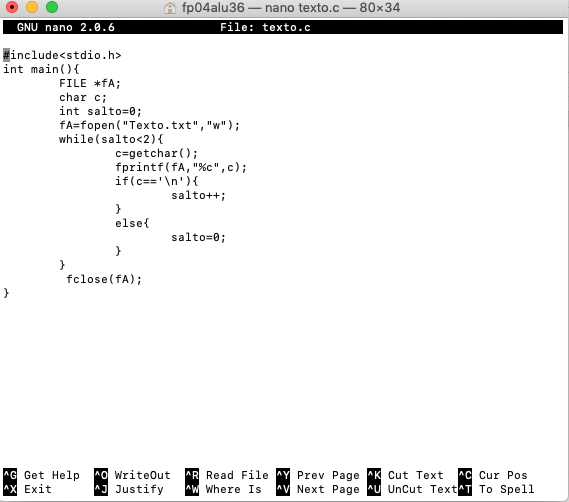
int fprintf(FILE \*apuntArch, char \*format,…);

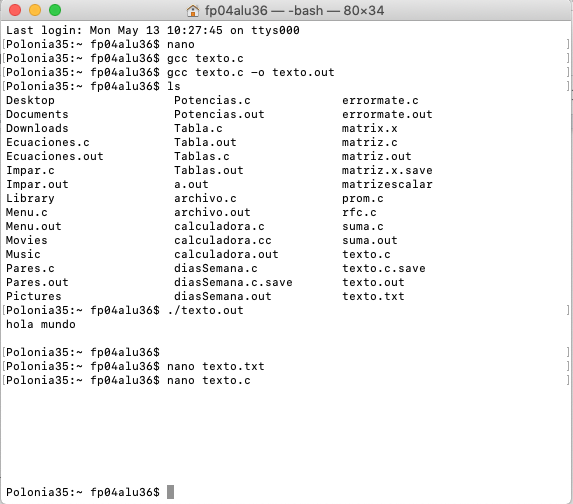
Donde apuntArch es un apuntador al archivo devuelto por una llamada a la función fopen(), es decir, fprintf() y fsanf() dirigen sus operaciones de E/S al archive al que apunta apuntArch. formato es una cadena que puede incluir texto o especificadores de impresión de variables.

En los puntos suspensivos se agregan las variables (si es que existen) que se quieran mandar a imprimir al archivo.

Actividades:

1. Código fuente para crear un archivo de tipo texto. Verificar el archivo creado

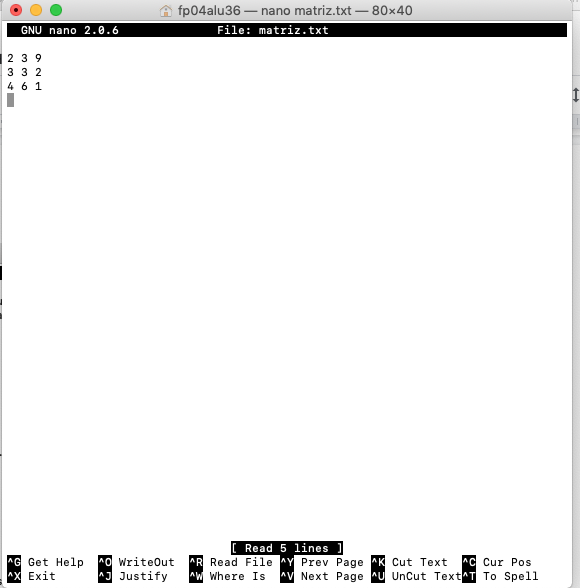


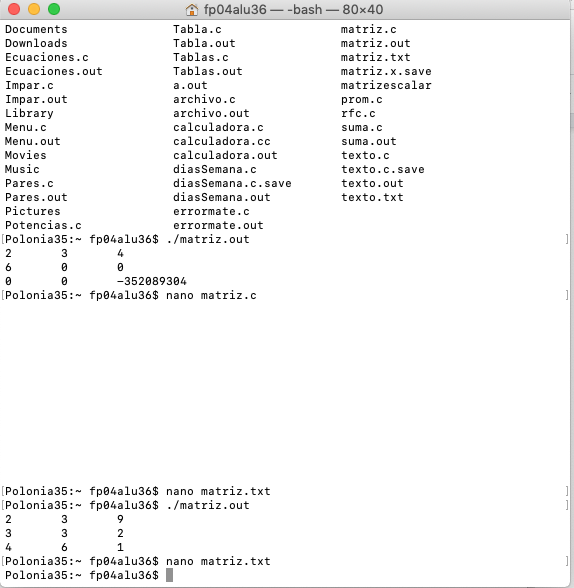




1. Código fuente para leer desde un archivo la matriz M de 3x3 de tipo entero. Mostrar en pantalla la matriz M.







Conclusiones:

Podemos concluir que la lectura y escritura de datos nos ayuda a reducir los códigos que realizamos a la hora de programar y que además nos ayudan como medio de almacenamiento secundario.

La lectura de archivos con datos de entrada, y la escritura de archivos de salida con los resultados, constituye una manera simple pero efectiva de comunicación.