**Trabajo de investigación de archivos**

**Nombre y apellido: Ramiro Ridolfi.**

Buscar, definir y ejemplificar los siguientes conceptos adjuntando las imágenes correspondientes

a) Crear un archivo en C

b) Abrir y cerrar un archivo

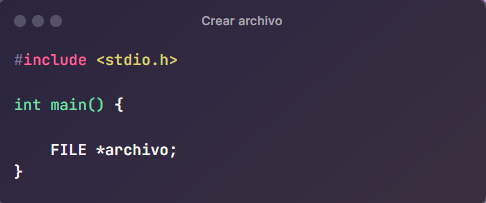
c) Leer, escribir, sobre escribir, etc. (parámetros)

d) Moverse dentro del fichero y escribir la palabra “PRINCIPIO” ,“ACTUAL” O “FINAL” según corresponda.

e) Diferencia entre fputc y fputs

a)

El estándar de C contiene varias funciones para la edición de ficheros, éstas están definidas en la cabecera stdio.h y por lo general empiezan con la letra f, haciendo referencia a file. Adicionalmente se agrega un tipo FILE, el cual se usará como apuntador a la información del fichero.



b)

La función **fopen** sirve para abrir y crear ficheros en disco.

El prototipo correspondiente de fopen es:



**Los parámetros de entrada de fopen son:**

filename: una cadena que contiene un nombre de fichero válido. opentype: especifica el tipo de fichero que se abrirá o se creará.

**Una lista de parámetros opentype para la función fopen:**

"r": abrir un archivo para lectura, el fichero debe existir.

"w": abrir un archivo para escritura, se crea si no existe o se sobrescribe si existe.

"a": abrir un archivo para escritura al final del contenido, si no existe se crea.

"r+": abrir un archivo para lectura y escritura, el fichero debe existir.

"w+": crear un archivo para lectura y escritura, se crea si no existe o se sobrescribe si existe.

"r+b" "rb+": Abre un archivo en modo binario para actualización (lectura y escritura).

"rb": Abre un archivo en modo binario para lectura.



La función **fclose** sirve para poder cerrar un fichero que se ha abierto.

El prototipo correspondiente de fclose es:



Un valor de retorno cero indica que el fichero ha sido correctamente cerrado, si ha habido algún error, el valor de retorno es la constante EOF.



c)

Un archivo generalmente debe verse como un string (una cadena de caracteres) que está guardado en el disco duro. Para trabajar con los archivos existen diferentes formas y diferentes funciones.

**FGETC**

La función **fgetc** lee un carácter a la vez del archivo que está siendo señalado con el puntero \*archivo. En caso de que la lectura sea exitosa devuelve el carácter leído y en caso de que no lo sea o de encontrar el final del archivo devuelve EOF.

El prototipo correspondiente de fgetc es:





**FGETS**

La función **fgets** está diseñada para leer cadenas de caracteres. Leerá hasta n-1 caracteres o hasta que lea un cambio de línea '\n' o un final de archivo EOF. En este último caso, el carácter de cambio de línea '\n' también es leído.

El prototipo correspondiente de fgets es:



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 función fgets. Y por último el puntero del archivo por supuesto que es la forma en que fgets sabrá a que archivo debe leer.



**fgets** es útil cuando quieres leer líneas completas de un archivo y manejarlas como cadenas, mientras que **fgetc** es más adecuada para leer caracteres individualmente.  
**EOF (End of File)** es un indicador especial utilizado en programación para señalar el final de un archivo. Se representa como una constante predefinida que indica el fin de los datos disponibles para leer en un archivo. Cuando un programa intenta leer más allá del EOF, generalmente se considera un indicador de que ya no hay más datos que leer en el archivo.

**FREAD**

La función **fread** 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 fgets 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.



**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:





Así como podemos leer datos desde un fichero, también se pueden crear y escribir ficheros con la información que deseamos almacenar.

**FPUTC**

La función **fputc** 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:





**FPUTS**

La función **fputs** escribe una cadena en un fichero. La ejecución de la misma no 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:





**FWRITE**

La función **fwrite** 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:





**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:





d)

**Para este programa, vamos a utilizar las siguientes funciones:**

**fopen:** Para abrir el archivo en modo lectura y escritura.

**fseek:** Para mover el puntero de posición dentro del archivo.

**Prototipo:** int fseek(FILE \*stream, long offset, int whence);

**fseek** se utiliza para mover el puntero de posición dentro de un archivo. El parámetro stream es el puntero al archivo, offset es el desplazamiento en bytes desde una posición de referencia determinada por whence, y whence indica la posición de referencia desde la cual se debe realizar el desplazamiento (SEEK\_SET, SEEK\_CUR, o SEEK\_END).

**ftell:** Para obtener la posición actual del puntero dentro del archivo.

**Prototipo:** long ftell(FILE \*stream);

**ftell** se utiliza para obtener la posición actual del puntero dentro de un archivo. El parámetro stream es el puntero al archivo.

**fprintf:** Para escribir las palabras "PRINCIPIO", "ACTUAL" o "FINAL" según corresponda en el archivo.

**fclose:** Para cerrar el archivo después de terminar de escribir en él.

**La lógica del programa será la siguiente:**

1 - Abrimos el archivo en modo lectura y escritura.

2 - Movemos el puntero al final del archivo usando **fseek**.

3 - Obtenemos la posición actual del puntero usando **ftell**.

4 - Si la posición es 0 (es decir, el archivo está vacío), escribimos "PRINCIPIO".

5 - Si la posición es diferente de 0 y menor que la posición actual del puntero (lo cual significa que hemos escrito algo previamente), escribimos "ACTUAL".

6 - Si la posición es igual a la posición actual del puntero (es decir, estamos al final del archivo), escribimos "FINAL".

7 - Cerramos el archivo.

e)

**fputc:** Esta función se utiliza para escribir un solo carácter en un archivo. Requiere especificar el carácter a escribir y el archivo donde se va a escribir.

**fputs:** Esta función se utiliza para escribir una cadena de caracteres (una cadena terminada en \0 o nulo) en un archivo. Requiere especificar la cadena de caracteres a escribir y el archivo donde se va a escribir.

**FUENTES DE LA INFORMACION:**

<https://w3.ual.es/~abecerra/ID/archivos>

<https://es.wikibooks.org/wiki/Programación_en_C/Manejo_de_archivos#>