


ARCHIVOS EN C

Cátedra Programación II

Octubre 2017

1. Apertura de un archivo

Se puede usar la función **fopen** para crear un nuevo archivo o abrir un archivo existente. Esta llamada a la función, inicializa un objeto del tipo **FILE**, el cual contiene toda la información necesaria para controlar la entrada/salida del archivo. El prototipo de esta llamada a función es:



```
FILE *fopen( const char * filename, const char * mode );
```


en este caso:

- el parámetro **filename** es un string que hace referencia al archivo. En caso que se encuentre en un directorio distinto del que estamos ejecutando tenemos que indicarle el path completo desde la raíz hasta la ubicación del mismo y, su nombre y extensión. En el caso que se encuentre en el directorio en el que nos encontramos, sólo es preciso poner el nombre y la extensión.
- el parámetro **mode** hace referencia al modo de acceso que puede tomar los siguientes valores:

Modo	Descripción
r	Abre un archivo existente para lectura
w	Abre un archivo para escritura. Si no existe, un nuevo archivo es creado. Se comienza a escribir desde el inicio del archivo, pisando la información existente, si es que tenía alguna.
a	Abre un archivo para escritura en formato append. Si no existe, un nuevo archivo es creado. Aquí el programa comienza a escribir desde el final del contenido que el archivo tenga, si es que tiene alguno.
r+	Abre un archivo para lectura y escritura.
w+	Abre un archivo para lectura y escritura. Si el archivo existe entonces pisa todo su contenido y, si no existe lo crea.
a+	Abre un archivo para lectura y escritura. Crea el archivo si no existe. La lectura va a comenzar del comienzo del archivo pero, la escritura puede ser sólo en formato append, es decir, al final del archivo.

2. Cierre de un archivo

Para cerrar un archivo se hace uso de la función **fclose**. El prototipo de la función es:




```
int fclose( FILE *fp );
```

la función **fclose** retorna cero si pudo cerrarse adecuadamente o **EOF** si existe un error al tratar de cerrar el archivo. Esta función es la que, al llamarse, vuelca el buffer con todos los datos pendientes al archivo, lo cierra y, libera la memoria usada por el archivo.

La constante EOF es definida en la librería **stdio.h**.


3. Escribiendo en un archivo

La forma más sencilla de escribir caracteres (de a uno!) es usando la función:




```
int fputc( int c, FILE *fp );
```

La función **fputc** escribe el caracter representado por el entero *c* al archivo al que apunta *fp*. Esta retorna **EOF** si la función tuvo algún error y, el entero que se quiso escribir si pudo realizar la operación.



```
int fputs( const char *s, FILE *fp );
```


La función **fputs** escribe el string *s* al archivo al que apunta *fp*. Se obtiene **EOF** si la función tuvo algún error y, un entero no negativo si pudo escribirse sin problemas. También puede usarse la función:



```
int fprintf(FILE *fp, const char *format, ...)
```

esta función es equivalente al **printf** que venimos usando para escribir en consola, siendo *format* el formato de escritura (**%d**, **%s**, **%c**, etc) y, a continuación, toma la lista de variables a escribir con el formato dado.

A continuación, podemos ver un ejemplo del uso de las funciones.




```
#include <stdio.h>

main() {
    FILE *fp;

    fp = fopen("test.txt", "w+");
    fprintf(fp, "This is testing for fprintf...\n");
    fputs("This is testing for fputs...\n", fp);
    fclose(fp);
}
```


4. Leyendo de un archivo

La forma más sencilla de leer caracteres (de a uno!) es usando la función:



```
int fgetc( FILE * fp );
```

La función **fgetc** lee un caracter del archivo al que apunta *fp*. Esta retorna **EOF** si la función tuvo algún error y, sino, el caracter leído. La siguiente función lee un string de un archivo:



```
char *fgets( char *buf, int n, FILE *fp );
```

La función **fgets** lee hasta $n - 1$ caracteres desde el archivo referenciado por *fp*. Este copia el string leído en el buffer **buf**, agregando un caracter `'\0'` al final del string. Si, durante la lectura, se encuentra un salto de línea, un `'\n'` o el final del archivo entonces, se leen la máxima cantidad de caracteres y, se retornan sólo los caracteres leídos hasta ese punto incluyendo el caracter de salto de línea.

También se puede usar la función:



```
int fscanf(FILE *fp, const char *format, ...)
```

esta función es equivalente al **scanf** que venimos usando para leer desde consola, siendo *format* el formato de escritura (**%d**, **%s**, **%c**, etc) y, a continuación, toma la lista de variables a escribir con el formato dado. Debemos recordar que esta función lee hasta encontrar un separador el cual puede ser un espacio en blanco, una tabulación, un salto de línea o el final del archivo.

A continuación, podemos ver un ejemplo del uso de las funciones.



```
#include <stdio.h>

main() {

    FILE *fp;
    char buff[255];

    fp = fopen("test.txt", "r");
    fscanf(fp, "%s", buff);
    printf("1 : %s\n", buff );

    fgets(buff, 255, (FILE*)fp);
    printf("2: %s\n", buff );

    fgets(buff, 255, (FILE*)fp);
    printf("3: %s\n", buff );
    fclose(fp);

}
```

El cual lee el archivo escrito en la sección anterior imprimiendo su resultado en consola.