	
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación

**Profesor:** Saavedraa Hernández Honorato

**Asignatura:** Fundamentos de programación

**Grupo:** 01

**No de Práctica(s):** 13 "Lectura y escritura de datos"

**Integrante(s):** Luis Salinas Ludwig

**Semestre:** 2018-1

**Fecha de entrega:** 20/11/2017

**Observaciones:**

CALIFICACIÓN: \_\_\_\_\_

salas A y B

## 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.

## Desarrollo

Tenemos memoria primaria que es la RAM y una memoria secundaria como discos duros, USB es decir todo lo que nos permite guardar nuestros archivos es más grande y más lenta y una de sus características es que cuando apagamos la computadora no se pierde la información en cambio en la memoria primaria si se pierde todo.

En la memoria secundaria se guarda en forma de un archivo. Un archivo es un conjunto de datos que se encuentra en la memoria secundaria y cada uno tiene un nombre. La extensión sirve para identificar el tipo de archivo con esto sabremos de que se trata, que se encuentra en el archivo y de que formato es. Un directorio puede contener archivos, así como tener subdirectorios que contendrán lo mismo.

Para comunicarnos con un archivo o realizamos con un apuntador de tipo FILE, si deseamos trabajar con un archivo necesito abrirlo después puedo leer o escribir y con el apuntador lo podremos mover ya que cada vez que leemos o escribimos el apuntador se mueve al finalizar tenemos que cerrar el archivo en este momento se escribe en el disco de memoria secundaria.

El apuntador de archivo se declara de la siguiente manera: FILE\*F , para poder utilizarlo necesito tener la librería studio.h.

## Abrir Archivo

La función fopen() abre una secuencia par que se pueda utilizar y asociarle un archivo vemos que su estructura es la siguiente:

```
*FILE fopen(char* nombre_archivo,char* modo)
```

Nosotros abriremos los archivos de modo texto, es decir que todo lo que leamos o escribamos el sistema espera que sea más que simple texto, si yo pongo algo raro que no sea texto no lo entenderá.

Los diferentes modos de apertura de archivos son los siguientes:

r: Abre un archivo de texto para lectura.

w: Crea un archivo de texto para escritura.

a: Abre un archivo de texto para añadir.

r+: Abre un archivo de texto para lectura / escritura.

w+: Crea un archivo de texto para lectura / escritura.

a+: Añade o crea un archivo de texto para lectura / escritura.

rb: Abre un archivo en modo lectura y binario.

wb: Crea un archivo en modo escritura y binario.

Cerrar archivo

La función `fclose()` cierra la secuencia que fue abierta mediante la llamada a `fopen()`.

La estructura o firma de esta función es la siguiente:

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

Posteriormente se vio el siguiente ejemplo de código (abrir cerrar archivo)

```

#include<stdio.h>

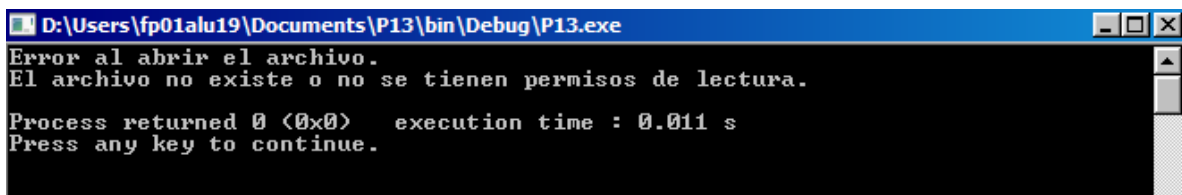
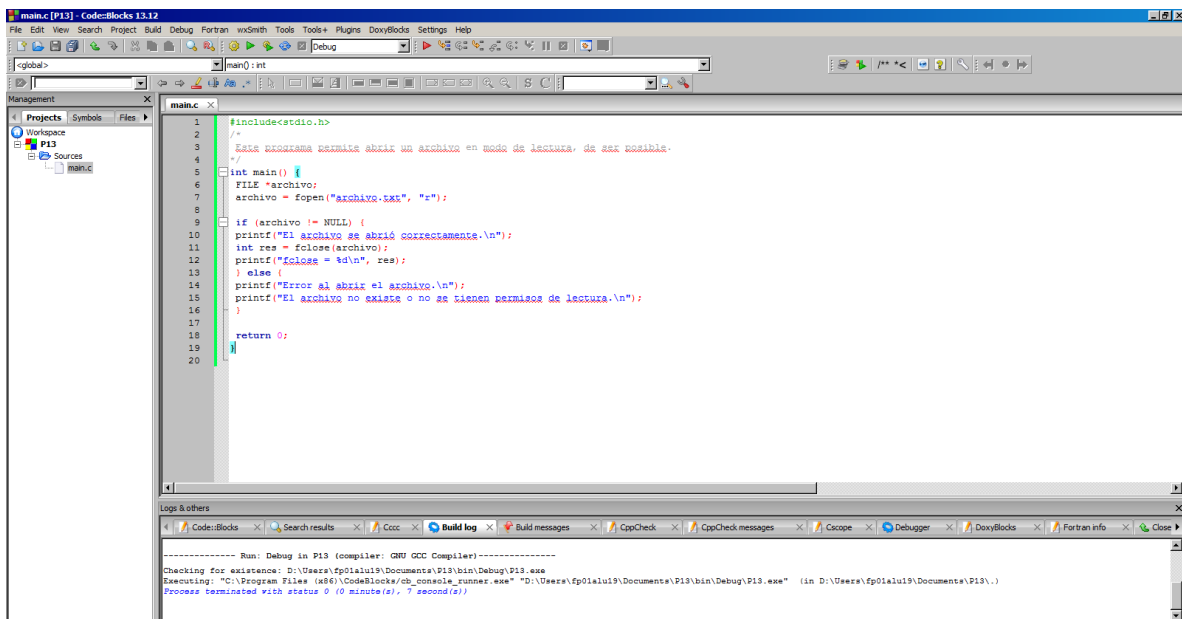
/*
    Este programa permite abrir un archivo en modo de lectura, de ser posible.
*/

int main() {
    FILE *archivo;
    archivo = fopen("archivo.txt", "r");

    if (archivo != NULL) {
        printf("El archivo se abrió correctamente.\n");
        int res = fclose(archivo);
        printf("fclose = %d\n", res);
    } else {
        printf("Error al abrir el archivo.\n");
        printf("El archivo no existe o no se tienen permisos de lectura.\n");
    }

    return 0;
}

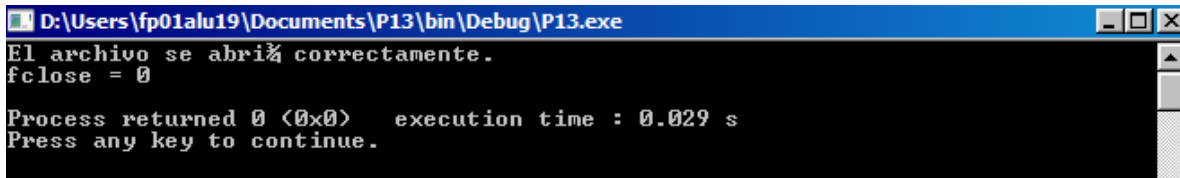
```



Si abro un archivo en modo escritura y dicho archivo no existe lo crear ya que la intención es escribir. Si fopen no puede abrir archivo me regresara NULL significa

que no se pudo abrir de lo contrario si el archivo lo pudo abrir nos imprimirá “El archivo se abrió correctamente” si esto no sucede ocurre el else donde nos imprimir “Error al abrir el archivo”, El archivo no existe o no se tienen permisos de lectura.

Par que sea cierto necesitamos crear un archivo con el nombre archio.txt con ello ya podremos ver el siguiente resultado



```
D:\Users\fp01alu19\Documents\P13\bin\Debug\P13.exe
El archivo se abrió correctamente.
fclose = 0
Process returned 0 (0x0) execution time : 0.029 s
Press any key to continue.
```

## Funciones fgets y fputs

Las funciones fgets() lee un archivo y fputs() escribe en un archivo, sus estructuras son las siguientes:

```
char *fgets(char *buffer, int tamaño, FILE *apArch);
```

```
char *fputs(char *buffer, FILE *apArch);
```

Al regresar un apuntador de tipo char fgets() permite leer una cadena desde un archivo específico y fputs() permite escribir una cadena en un archivo específico.

Se vio el siguiente ejemplo del código fgets

```

#include<stdio.h>

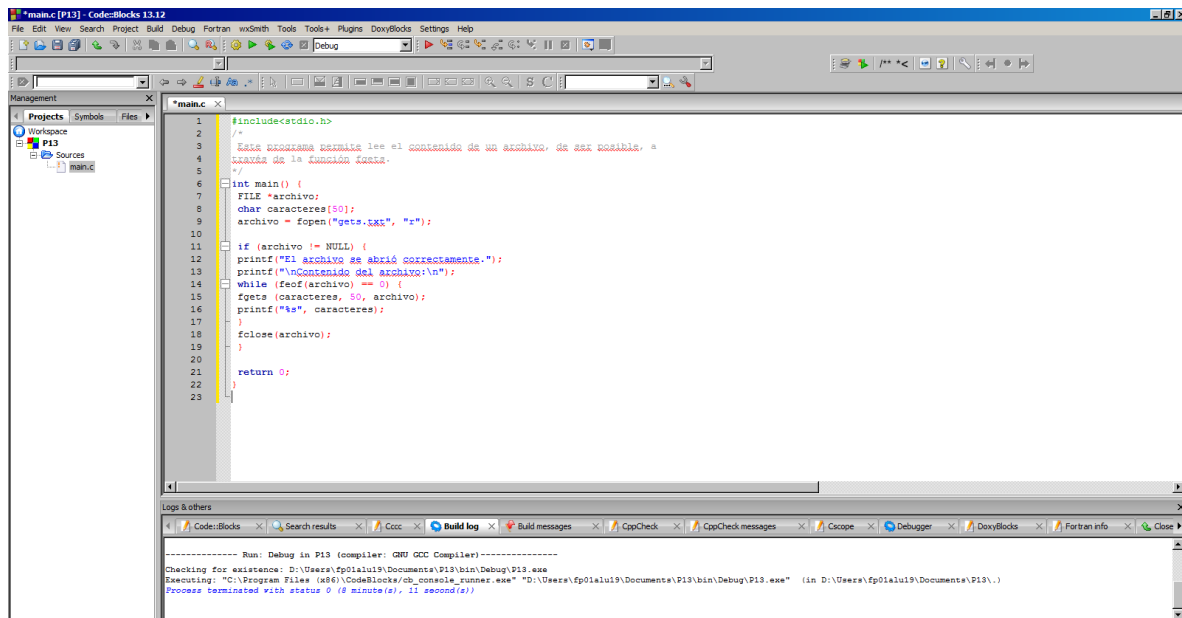
/*
    Este programa permite leer el contenido de un archivo, de ser posible, a
    través de la función fgets.
*/

int main() {
    FILE *archivo;
    char caracteres[50];
    archivo = fopen("gets.txt", "r");

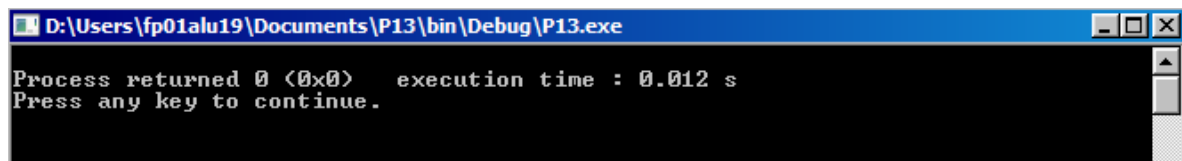
    if (archivo != NULL) {
        printf("El archivo se abrió correctamente.");
        printf("\nContenido del archivo:\n");
        while (feof(archivo) == 0) {
            fgets (caracteres, 50, archivo);
            printf("%s", caracteres);
        }
        fclose(archivo);
    }

    return 0;
}

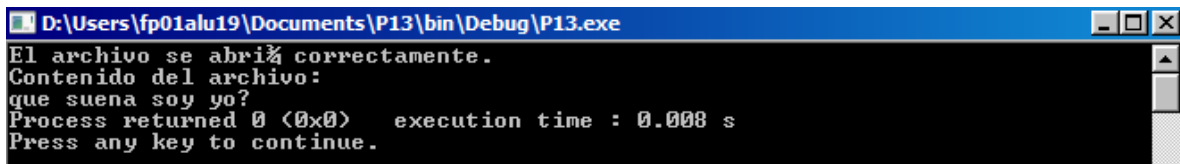
```



Como en el ejemplo anterior tenemos que crear el archivo para que nos logre imprimir y que si no o teneos el resultado será:



Al crear el archivo corre de manera correcta y nos imprime lo que está en el programa:



```
D:\Users\fp01alu19\Documents\P13\bin\Debug\P13.exe
El archivo se abrió correctamente.
Contenido del archivo:
que suena soy yo?
Process returned 0 (0x0) execution time : 0.008 s
Press any key to continue.
```

Con `fgets` yo puedo decidir el tamaño que deseo leer, pero en cambio `fputs` no se puede hacer esto.

Continuamos con el código `fputs`

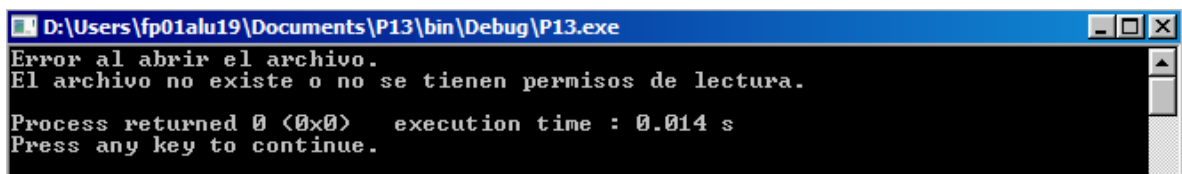
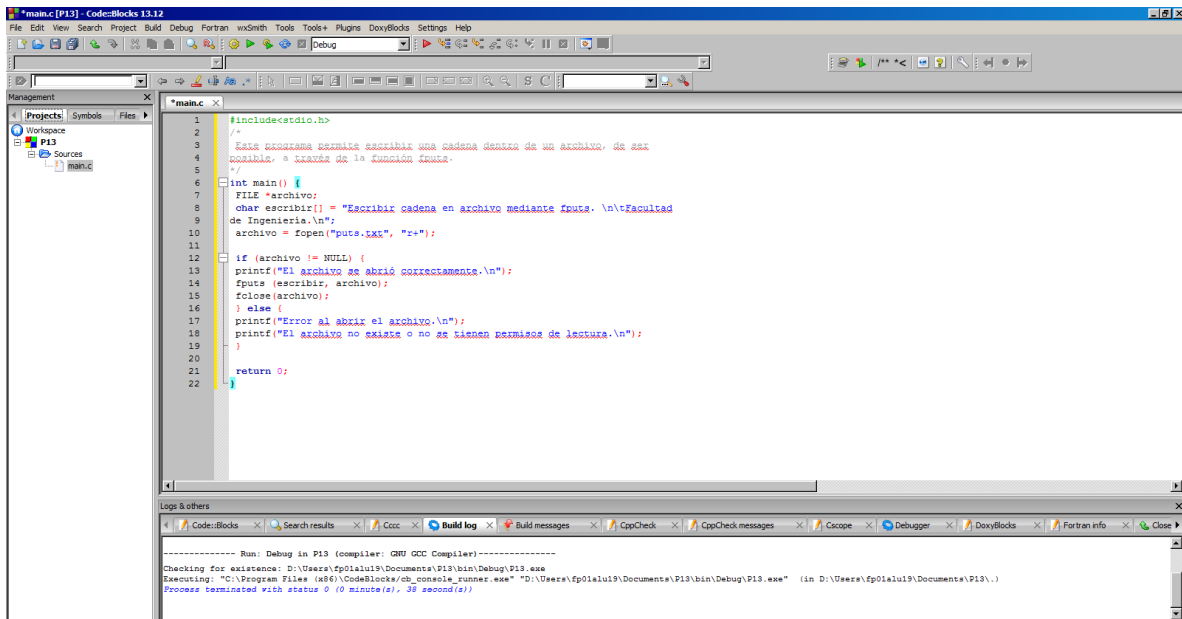
```
#include<stdio.h>

/*
    Este programa permite escribir una cadena dentro de un archivo, de ser
    posible, a través de la función fputs.
*/

int main() {
    FILE *archivo;
    char escribir[] = "Escribir cadena en archivo mediante fputs. \n\tFacultad
de Ingeniería.\n";
    archivo = fopen("puts.txt", "r+");

    if (archivo != NULL) {
        printf("El archivo se abrió correctamente.\n");
        fputs (escribir, archivo);
        fclose(archivo);
    } else {
        printf("Error al abrir el archivo.\n");
        printf("El archivo no existe o no se tienen permisos de lectura.\n");
    }

    return 0;
}
```



## Funciones fscanf y fprintf

Estas funciones se comportan exactamente como printf() que es imprimir y scanf() que es leer en excepción que opera sobre archivo. Las estructuras son las siguientes:

```
int fprintf(FILE *apArch, char *formato, ...);
```

```
int fscanf(FILE *apArch, char *formato, ...);
```

Continuamos con el código fscanf



```

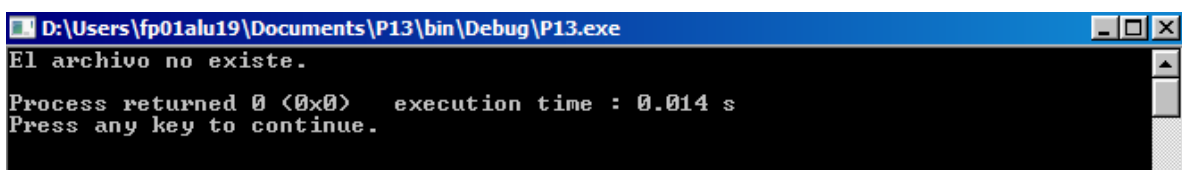
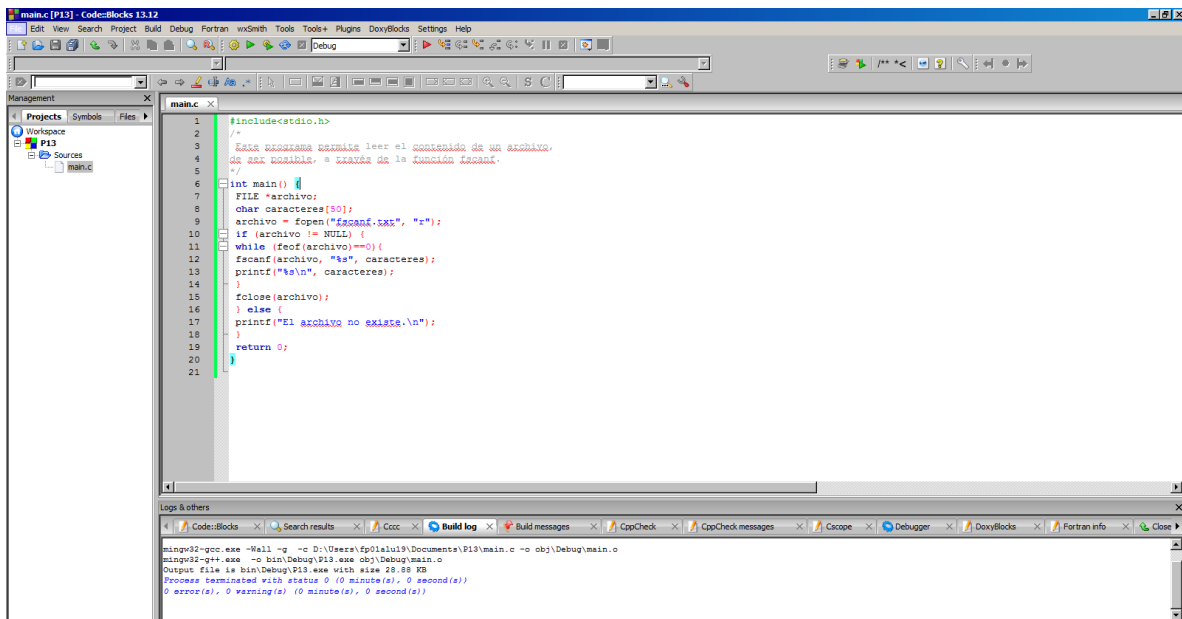
#include<stdio.h>

/*
    Este programa permite leer el contenido de un archivo,
    de ser posible, a través de la función fscanf.
*/

int main() {
    FILE *archivo;
    char caracteres[50];
    archivo = fopen("fscanf.txt", "r");
    if (archivo != NULL) {
        while (feof(archivo)==0){
            fscanf(archivo, "%s", caracteres);
            printf("%s\n", caracteres);
        }
        fclose(archivo);
    } else {
        printf("El archivo no existe.\n");
    }
    return 0;
}

```

El fscanf() está leyendo un cadena porque le decimos que lea una cadena y que lo indicamos con él %s

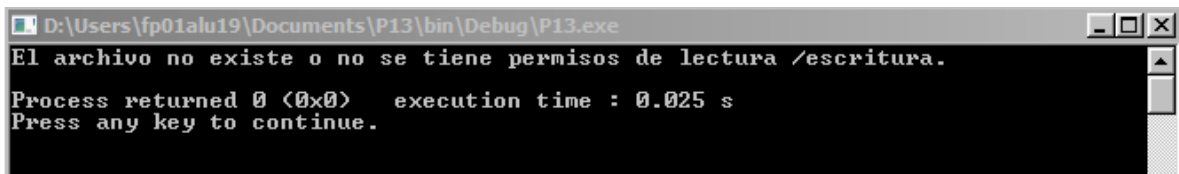
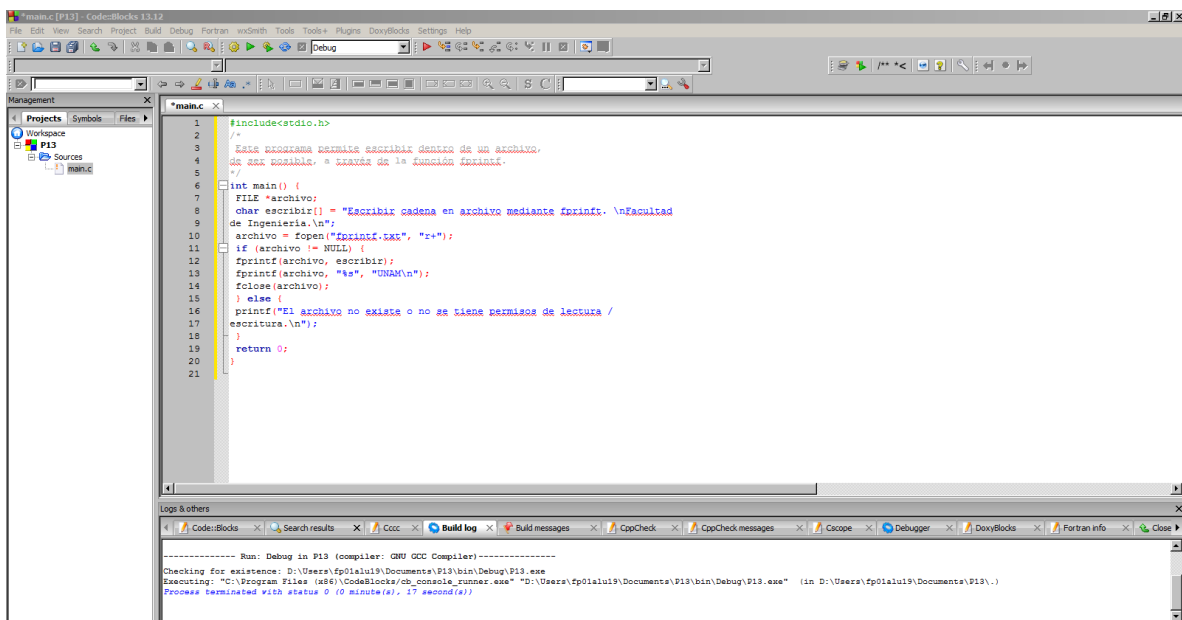


## Código fprintf

```
#include<stdio.h>

/*
    Este programa permite escribir dentro de un archivo,
    de ser posible, a través de la función fprintf.
*/

int main() {
    FILE *archivo;
    char escribir[] = "Escribir cadena en archivo mediante fprintf. \nFacultad
de Ingeniería.\n";
    archivo = fopen("fprintf.txt", "r+");
    if (archivo != NULL) {
        fprintf(archivo, escribir);
        fprintf(archivo, "%s", "UNAM\n");
        fclose(archivo);
    } else {
        printf("El archivo no existe o no se tiene permisos de lectura /
escritura.\n");
    }
    return 0;
}
```



## Funciones fread y fwrite

### Código fread

```
#include <stdio.h>

/*
    Este programa muestra el contenido de un archivo de texto. El
    nombre del archivo se recibe como argumento de la
    función principal.
*/

int main(int argc, char **argv) {
    FILE *ap;
    unsigned char buffer[2048]; // Buffer de 2 Kbytes
    int bytesLeidos;

    // Si no se ejecuta el programa correctamente
    if(argc < 2) {
        printf("Ejecutar el programa de la siguiente\nmanera:\n\tnombre_\tprograma nombre_archivo\n");
        return 1;
    }

    // Se abre el archivo de entrada en modo lectura y binario
    ap = fopen(argv[1], "rb");

    if(!ap) {
        printf("El archivo %s no existe o no se puede abrir", argv[1]);
        return 1;
    }

    while(bytesLeidos = fread(buffer, 1, 2048, ap))
        printf("%s", buffer);

    fclose(ap);

    return 0;
}
```

The screenshot displays the Code::Blocks IDE interface. The main editor window shows a C program named `main.c` with the following code:

```
3 Este programa muestra el contenido de un archivo de texto. El
4 nombre del archivo se recibe como argumento de la
5 función principal.
6 */
7
8 int main(int argc, char **argv) {
9     FILE *ap;
10    unsigned char buffer[2048]; // Buffer de 2048 bytes
11    int bytesLeidos;
12
13    // Si no se ejecuta el programa correctamente
14    if (argc < 2) {
15        printf("Ejecutar el programa de la siguiente manera:\n\tnombre_programa nombre_archivo\n");
16        return 1;
17    }
18
19    // Se abre el archivo de texto en modo lectura y binario
20    ap = fopen(argv[1], "rb");
21    if (!ap) {
22        printf("El archivo %s no existe o no se puede abrir", argv[1]);
23        return 1;
24    }
25
26    while (bytesLeidos = fread(buffer, 1, 2048, ap))
27        printf("%s", buffer);
28
29    fclose(ap);
30    return 0;
31 }
```

Below the editor, the 'Log & others' panel shows build messages:

```
==== Build: Debug in P13 (compiler: GNU GCC Compiler) ====
In function 'main':
D:\Users\fp01a... 28 warning: suggest parentheses around assignment used as truth value [-Wparentheses]
==== Build finished: 0 error(s), 1 warning(s) (0 minute(s), 0 second(s)) ====
==== Run: Debug in P13 (compiler: GNU GCC Compiler) ====
```

At the bottom, a terminal window titled `D:\Users\fp01a19\Documents\P13\bin\Debug\P13.exe` shows the program's output:

```
Ejecutar el programa de la siguiente manera:
nombre_programa nombre_archivo

Process returned 1 (0x1)   execution time : 0.012 s
Press any key to continue.
```

Código fwrite

```

#include <stdio.h>

/*
    Este programa realizar una copia exacta de dos archivos. Los
    nombres de los archivos (origen y destino) se reciben como
    argumentos de la función principal.
*/

int main(int argc, char **argv) {
    FILE *archEntrada, *archivoSalida;
    unsigned char buffer[2048]; // Buffer de 2 Kbytes
    int bytesLeidos;

    // Si no se ejecuta el programa correctamente
    if(argc < 3) {
        printf("Ejectuar el programa de la siguiente manera:\n");
        printf("\tnombre_programa \tarchivo_origen \tarchivo_destino\n");
        return 1;
    }

    // Se abre el archivo de entrada en modo de lectura y binario
    archEntrada = fopen(argv[1], "rb");

    if(!archEntrada) {
        printf("El archivo %s no existe o no se puede abrir", argv[1]);
        return 1;
    }

    // Se crea o sobrescribe el archivo de salida en modo binario
    archivoSalida = fopen(argv[2], "wb");

    if(!archivoSalida) {
        printf("El archivo %s no puede ser creado", argv[2]);
        return 1;
    }

    // Copia archivos
    while (bytesLeidos = fread(buffer, 1, 2048, archEntrada))
        fwrite(buffer, 1, bytesLeidos, archivoSalida);

    // Cerrar archivos
    fclose(archEntrada);
    fclose(archivoSalida);

    return 0;
}

```

