



**Universidad Nacional  
Autónoma de México**  
Facultad de Ingeniería



**Fundamentos de Programación (1122)**

Laboratorios de computación  
salas A y B

*Profesor: M.I. Marco Antonio Martínez Quintana*  
*Semestre 2021-1*

**Practica No. 13**

Lectura y escritura de datos

Grupo: 1129

No. de Equipo de cómputo empleado: No aplica

No. de Lista o Brigada: No aplica

No. de Lista: 42

**Nombre: Adolfo Román Jiménez**

## **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.

## **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.

Los archivos nos permiten almacenar datos, de tal forma que nuestra información ocupe un determinado espacio en el disco duro de nuestra computadora, permitiendo conservar la información después de la ejecución de un programa.

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.

## **Apuntador a archivo**

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

Un apuntador a 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 cabecera 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 la siguiente:

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

Donde nombre\_archivo es un puntero a 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 cómo se abre el archivo.

Existen diferentes modos de apertura de archivos, los cuales se mencionan a continuación, además de que se pueden utilizar más de uno solo:

- 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 una secuencia que fue abierta mediante una llamada a `fopen()`. Escribe la información que se encuentre en el buffer al disco y 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 pérdida de datos, destrucción de archivos y posibles errores intermitentes en el programa. La firma de esta función es:

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

Donde `apArch` es el apuntador al archivo devuelto por la llamada a `fopen()`. Si se devuelve un valor cero significa que la operación de cierre ha tenido éxito. Generalmente, esta función solo falla cuando un disco se ha retirado antes de tiempo o cuando no queda espacio libre en el mismo.

- 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;
}
```

```
1 #include<stdio.h>
2
3 /*
4 Este programa permite abrir un archivo en modo de lectura, de ser posible.
5 */
6
7 int main()
8 {
9     FILE *archivo;
10
11     archivo = fopen("archivo.txt", "r");
12
13     if (archivo != NULL)
14     {
15         printf("El archivo se abrió correctamente.\n");
16         int res = fclose(archivo);
17         printf("fclose = %d\n", res);
18     }
19     else
20     {
21         printf("Error al abrir el archivo.\n");
22         printf("El archivo no existe o no se tienen permisos de lectura.\n");
23     }
24
25     return 0;
26 }
27
```

```
~/ x +
~/ $ clear
~/ $ clang -o test practica13.c
~/ $ ./test
El archivo se abrió correctamente.
fclose = 0
~/ $
```

## Funciones fgets y puts

Las funciones fgets() y fputs() pueden leer y escribir, respectivamente, cadenas sobre los archivos. Las firmas de estas funciones son, respectivamente:

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

La función fputs() permite escribir una cadena en un archivo específico. La función fgets() permite leer una cadena desde el archivo especificado. Esta función lee un renglón a la vez.

### - 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;  
}
```

```

1 #include<stdio.h>
2
3 /*
4 Este programa permite lee el contenido de un
5 archivo, de ser posible, a través de la función fgets.
6 */
7
8 int main()
9 {
10     FILE *archivo;
11     char caracteres[50];
12     archivo = fopen("gets.txt", "r");
13
14     if (archivo != NULL)
15     {
16         printf("El archivo se abrió correctamente.");
17         printf("\nContenido del archivo:\n");
18
19         while (feof(archivo) == 0)
20         {
21             fgets (caracteres, 50, archivo);
22             printf("%s", caracteres);
23         }
24
25         fclose(archivo);
26     }
27
28     return 0;
29 }

```

```

~/ $ clang -o test practica13.c
~/ $ ./test
El archivo se abrió correctamente.
Contenido del archivo:
D' You Know What I Mean?

Oasis

Step off the train all alone at dawn
Back into the hole where I was born
The sun in the sky never raised an eye to me
The blood on the tracks, and they must be mine
The fool on the hill, and I feel fine
Don't look back 'cause you know what you might see
Look into the wall of my mind's eye
I think I know, but I don't know why
The questions are the answers you might need
Coming in a mess, going out in style
I ain't good-looking, but I'm someone's child
No one can give me the air that's mine to breatheNo one can give me the air that's mine to breathe~/ $ 

```

## - 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;
}
```

```
1 #include<stdio.h>
2
3 /*
4  Este programa permite escribir una cadena dentro de un archivo, de ser posible, a través de la función fputs.
5  */
6
7 int main()
8 {
9     FILE *archivo;
10    char escribir[] = "Escribir cadena en archivo mediante fputs. \n\tFacultad de Ingeniería.\n";
11
12    archivo = fopen("puts.txt", "r+");
13
14    if (archivo != NULL)
15    {
16        printf("El archivo se abrió correctamente.\n");
17        fputs (escribir, archivo);
18        fclose(archivo);
19    }
20    else
21    {
22        printf("Error al abrir el archivo.\n");
23        printf("El archivo no existe o no se tienen permisos de lectura.\n");
24    }
25    return 0;
26 }
27
28
```

```
~/ $ clang -o test practica13.c
~/ $ ./test
El archivo se abrió correctamente.
~/ $
```

```
puts.txt  practica13.c
1 Escribir cadena en archivo mediante fputs.
2 Facultad de Ingeniería.
3
```

## Funciones fscanf y fperintf

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

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

Donde apArch es un apuntador al archivo devuelto por una llamada a la función fopen(), es decir, fprintf() y fscanf() dirigen sus operaciones de E/S al archivo al que apunta apArch. 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) cuyos valores se quieren escribir en el archivo.

### - 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;
}
```



## - Código

```
1 #include<stdio.h>
2 /*
3 Este programa permite leer el contenido de un archivo, de ser posible, a través de la función fscanf.
4 */
5 int main()
6 {
7     FILE *archivo;
8     char caracteres[50];
9     archivo = fopen("fscanf.txt", "r");
10    if (archivo != NULL)
11    {
12        while (feof(archivo)==0)
13        {
14            fscanf(archivo, "%s", caracteres);
15            printf("%s\n", caracteres);
16        }
17
18        fclose(archivo);
19    }
20    else
21    {
22        printf("El archivo no existe.\n");
23    }
24
25    return 0;
26 }
27
```

```
~/ $ clear
~/ $ clang -o test practica13.c
~/ $ ./test
D'
You
Know
What
I
Mean?
Oasis
Step
off
the
train
all
alone
at
dawn
Back
into
the
hole
where
I
was
born
```

```
1 D' You Know What I Mean?
2
3 Oasis
4
5 Step off the train all alone at dawn
6 Back into the hole where I was born
7 The sun in the sky never raised an eye to me
8 The blood on the tracks, and they must be mine
9 The fool on the hill, and I feel fine
10 Don't look back 'cause you know what you might see
11 Look into the wall of my mind's eye
12 I think I know, but I don't know why
13 The questions are the answers you might need
14 Coming in a mess, going out in style
15 I ain't good-looking, but I'm someone's child
16 No one can give me the air that's mine to breathe
```

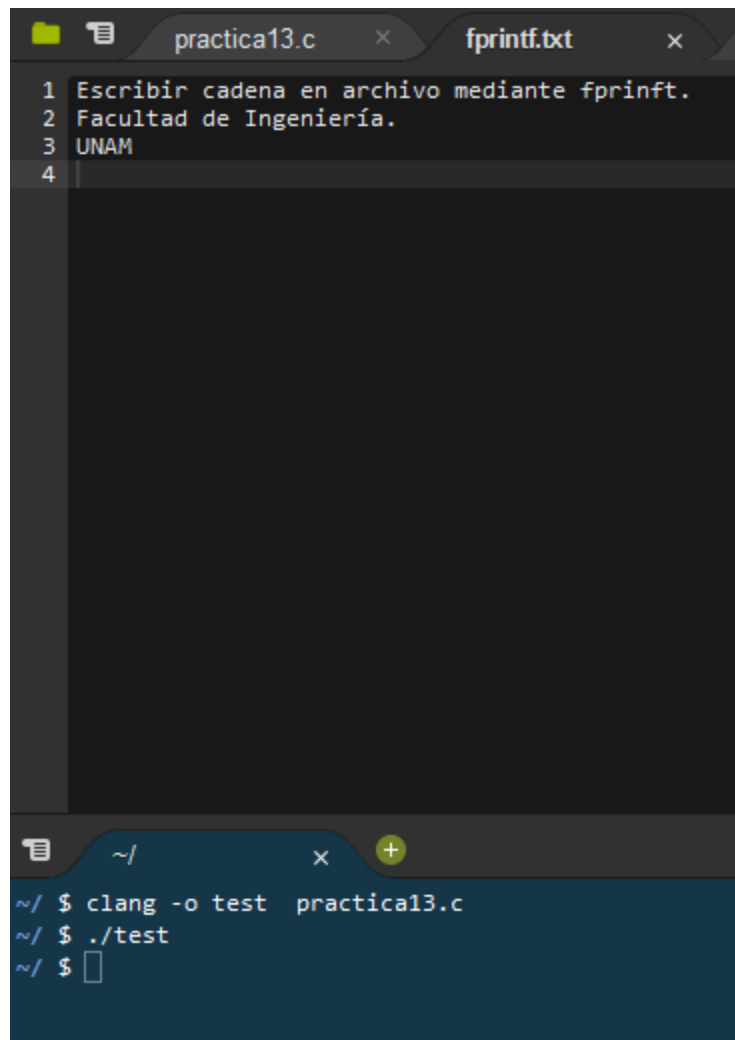
#### - Código (fprintf)

```
#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;
}
```

```

1 #include<stdio.h>
2 /*
3 Este programa permite escribir dentro de un archivo, de ser posible, a través de la función fprintf.
4 */
5 int main()
6 {
7     FILE *archivo;
8     char escribir[] = "Escribir cadena en archivo mediante fprintf. \nFacultad de Ingeniería.\n";
9     archivo = fopen("fprintf.txt", "r+");
10
11     if (archivo != NULL)
12     {
13         fprintf(archivo, "%s", escribir);
14         fprintf(archivo, "%s", "UNAM\n");
15         fclose(archivo);
16     }
17     else
18     {
19         printf("El archivo no existe o no se tiene permisos de lectura / escritura.\n");
20     }
21
22     return 0;
23 }
24

```



The image shows a code editor with two tabs: 'practica13.c' and 'fprintf.txt'. The 'fprintf.txt' tab is active and displays the output of the program:

```

1 Escribir cadena en archivo mediante fprintf.
2 Facultad de Ingeniería.
3 UNAM
4

```

Below the code editor is a terminal window with the following commands and output:

```

~/ $ clang -o test practica13.c
~/ $ ./test
~/ $

```

## Funciones fread y fwrite

fread y fwrite son funciones que permiten trabajar con elementos de longitud conocida. fread permite leer uno o varios elementos de la misma longitud a partir de una dirección de memoria determinada (apuntador).

El valor de retorno es el número de elementos (bytes) leídos. Su sintaxis es la siguiente:

```
int fread(void *ap, size_t tam, size_t nelem, FILE *archivo)
```

fwrite permite escribir hacia un archivo uno o varios elementos de la misma longitud almacenados a partir de una dirección de memoria determinada.

El valor de retorno es el número de elementos escritos. Su sintaxis es la siguiente:

```
int fwrite(void *ap, size_t tam, size_t nelem, FILE *archivo)
```

### - 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
                manera:\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;
}

```

```

1 #include <stdio.h>
2
3 /*
4 Este programa muestra el contenido de un archivo de texto.
5 El nombre del archivo se recibe como argumento de la
6 función principal.
7 */
8
9 int main(int argc, char **argv)
10 {
11     FILE *ap;
12
13     unsigned char buffer[2048]; // Buffer de 2 Kbytes
14     int bytesLeidos;
15     // Si no se ejecuta el programa correctamente
16     if(argc < 2)
17     {
18         printf("Ejecutar el programa de la siguiente manera:\n\tnombre\tprograma nombre_archivo\n");
19         return 1;
20     }
21
22     // Se abre el archivo de entrada en modo lectura y binario
23
24     ap = fopen(argv[1], "rb");
25
26     if(!ap)
27     {
28         printf("El archivo %s no existe o no se puede abrir", argv[1]);
29         return 1;
30     }
31
32     while(bytesLeidos == fread(buffer, 1, 2048, ap))
33     {
34         printf("%s", buffer);
35         fclose(ap);
36         return 0;
37     }
38 }

```

```
~/ $ clang -o test practica13.c
~/ $ ./test
Ejecutar el programa de la siguiente manera:
    nombre_ programa nombre_archivo
~/ $ clear
~/ $ clang -o test practica13.c
~/ $ ./test prueba.txt resultado.txt
~/ $ ./test prueba.txt resultado.txt
~/ $ ./test prueba.txt
~/ $ ./test prueba.txt
~/ $ ./test prueba.txt resultado.txt
~/ $ ./test
Ejecutar el programa de la siguiente manera:
    nombre_ programa nombre_archivo
~/ $ ./test prueba
El archivo prueba no existe o no se puede abrir~/ $ ./test prueba.txt
~/ $
```

Tuve problemas con este código, no entendí muy bien porque no funcionaba.

#### - 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("Ejecutar 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;
}
```

```

1 #include <stdio.h>
2
3 /*
4  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.
5  */
6
7 int main(int argc, char **argv)
8 {
9     FILE *archEntrada, *archivoSalida;
10
11     unsigned char buffer[2048]; // Buffer de 2 Kbytes
12     int bytesLeidos;
13     // Si no se ejecuta el programa correctamente
14
15     if(argc < 3)
16     {
17         printf("Ejecutar el programa de la siguiente manera:\n");
18         printf("\tnombre_programa \tarchivo_origen \tarchivo_destino\n");
19         return 1;
20     }
21     // Se abre el archivo de entrada en modo de lectura y binario
22
23     archEntrada = fopen(argv[1], "rb");
24
25     if(!archEntrada)
26     {
27         printf("El archivo %s no existe o no se puede abrir", argv[1]);
28         return 1;
29     }
30
31     // Se crea o sobrescribe el archivo de salida en modo binario
32
33     archivoSalida = fopen(argv[2], "wb");
34
35     if(!archivoSalida)
36     {
37         printf("El archivo %s no puede ser creado", argv[2]);
38         return 1;
39     }
40
41     // Copia archivos
42     while ((bytesLeidos = fread(buffer, 1, 2048, archEntrada)) > 0) fwrite(buffer, 1, bytesLeidos, archivoSalida);
43
44     // Cerrar archivos
45     fclose(archEntrada);
46     fclose(archivoSalida);
47
48     return 0;
49 }

```

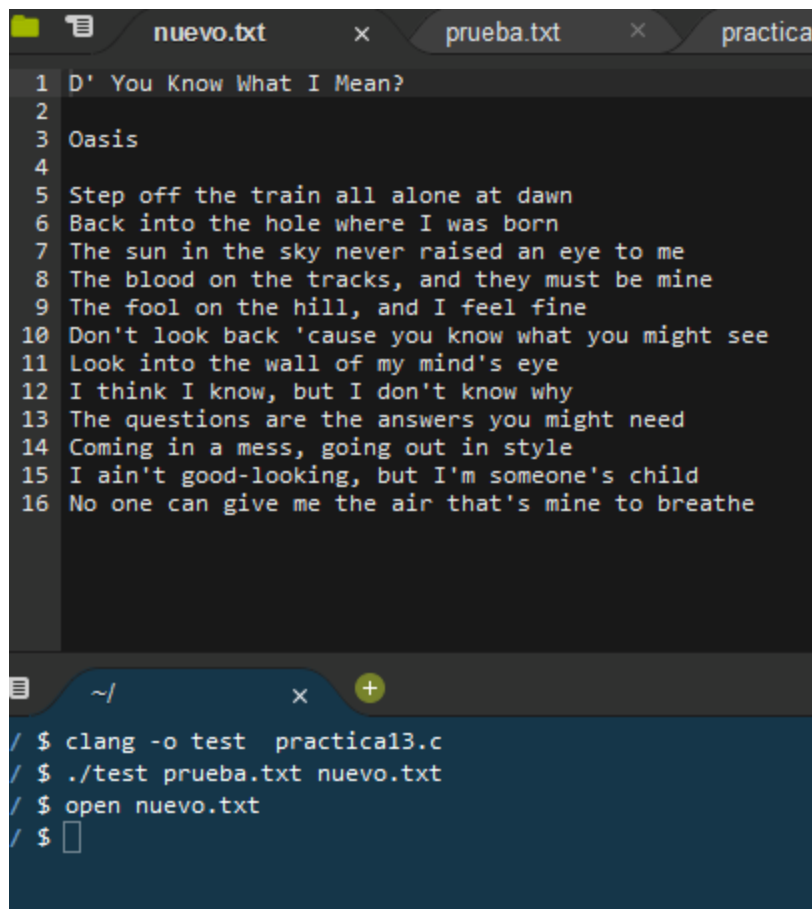
prueba.txt x practica13.c x fprint

```

1 D' You Know What I Mean?
2
3 Oasis
4
5 Step off the train all alone at dawn
6 Back into the hole where I was born
7 The sun in the sky never raised an eye to me
8 The blood on the tracks, and they must be mine
9 The fool on the hill, and I feel fine
10 Don't look back 'cause you know what you might see
11 Look into the wall of my mind's eye
12 I think I know, but I don't know why
13 The questions are the answers you might need
14 Coming in a mess, going out in style
15 I ain't good-looking, but I'm someone's child
16 No one can give me the air that's mine to breathe

```



The image shows a screenshot of a code editor and a terminal window. The code editor at the top has three tabs: 'nuevo.txt', 'prueba.txt', and 'practica'. The 'prueba.txt' tab is active, displaying the lyrics of the song 'D'You Know What I Mean?' by Oasis, numbered 1 through 16. The terminal window at the bottom shows a series of commands: compiling 'practical3.c' with clang, running the resulting 'test' program with 'prueba.txt' and 'nuevo.txt' as arguments, opening 'nuevo.txt', and a prompt for further input.

```
1 D' You Know What I Mean?
2
3 Oasis
4
5 Step off the train all alone at dawn
6 Back into the hole where I was born
7 The sun in the sky never raised an eye to me
8 The blood on the tracks, and they must be mine
9 The fool on the hill, and I feel fine
10 Don't look back 'cause you know what you might see
11 Look into the wall of my mind's eye
12 I think I know, but I don't know why
13 The questions are the answers you might need
14 Coming in a mess, going out in style
15 I ain't good-looking, but I'm someone's child
16 No one can give me the air that's mine to breathe

~/
/ $ clang -o test practical3.c
/ $ ./test prueba.txt nuevo.txt
/ $ open nuevo.txt
/ $
```

## Conclusiones:

La practica me gusto mucho porque yo esperaba poder manejar este tema cuando hice mi proyecto final, con el cual esperaba poder transmitir información desde un archivo .txt y generar un arreglo a partir de eso que diseminara la información de acuerdo a como el usuario la necesitaba.

Lamentablemente en ese momento estas funciones como fgets or fscanf eran algo nuevas para mi y aunque aun sigo sin dominarlas apropiadamente, espero que sin ningún problema, las pueda manejar en un futuro no muy lejano.

La practica fue en general sencilla, salvo por una excepción por ahí donde realmente no supe porque no estaba compilando el código adecuadamente.

## Referencias:

- Facultad de ingeniería - UNAM. Manual de prácticas del laboratorio de Fundamentos de programación: [http://odin.fi-b.unam.mx/salac/practicasFP/MADO-17\\_FP.pdf](http://odin.fi-b.unam.mx/salac/practicasFP/MADO-17_FP.pdf)