



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Profa. Karina Garcia Morales

Asignatura: Fundamentos de la programación.

Grupo: 22

No. de práctica(s): 12

Integrante(s): 1

No. de lista o brigada: 14

Semestre: Primero

Fecha de entrega:

Observaciones:

CALIFICACIÓN: _____

p12: Lectura y escritura de datos

OBJETIVO: El alumno elaborará 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.

CONCEPTOS:

*Sintaxis de funciones.

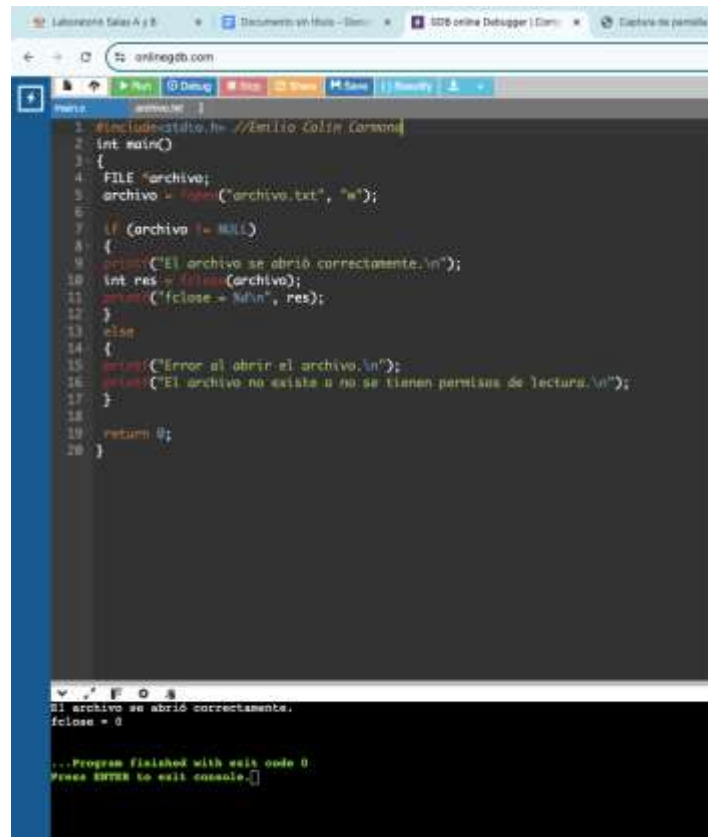
- `tipoValorRetorno nombre (parámetros)`
`{`
`// bloque de código de la función`
`}`

*Funciones de lectura y escritura en un archivo.

- La función `fputs()` permite escribir una cadena en un archivo específico. La función `fputs()` copia *serie* en la salida *ruta* en la posición actual. No copia el carácter nulo (`\0`) al final de la serie.
- La función `fgets()` permite leer una cadena desde el archivo especificado. Esta función lee un renglón a la vez. `fgets()` lee caracteres de la posición actual de *ruta* hasta el primer carácter de nueva línea (`\n`), hasta el final de la ruta o hasta que el número de caracteres leídos sea igual a *n*-1, lo que ocurra primero. La función `fgets()` almacena el resultado en *serie* y añade un carácter nulo (`\0`) al final de la serie. La *serie* incluye el carácter de nueva línea, si se lee. Si *n* es igual a 1, la *serie* está vacía.

ACTIVIDAD

1- cambiar el programa1.c para que ahora sí cree el archivo, esto se hace cambiando el “r” de read en el `fopen` por “w” de write.



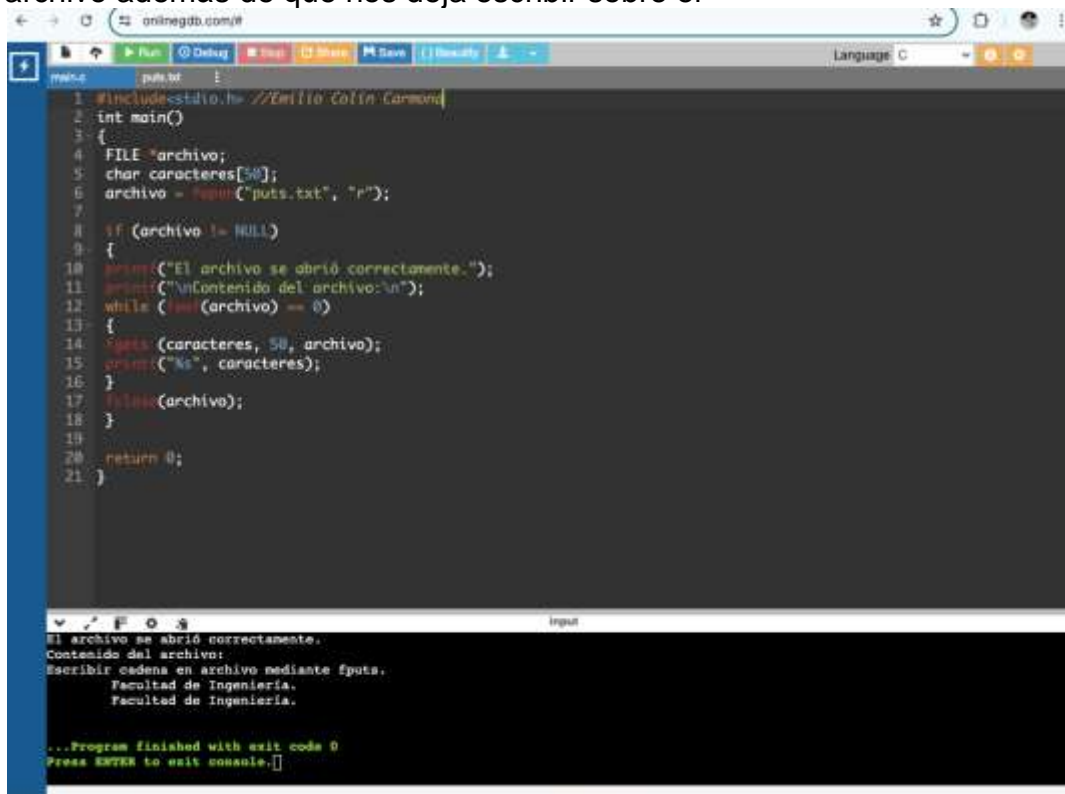
```
1 #include <stdio.h> //Emilio Colin Corvino
2 int main()
3 {
4     FILE *archivo;
5     archivo = fopen("archivo.txt", "w");
6
7     if (archivo != NULL)
8     {
9         printf("El archivo se abrió correctamente.\n");
10        int res = fgetc(archivo);
11        printf("fclose = %d\n", res);
12    }
13    else
14    {
15        printf("Error al abrir el archivo.\n");
16        printf("El archivo no existe o no se tienen permisos de lectura.\n");
17    }
18
19    return 0;
20 }
```

El archivo se abrió correctamente.
fclose = 0

...Program finished with exit code 0
Press ENTER to exit console.

2- corro primero el programa3.c para que abra un archivo con `fputs` y después lo reemplazo con el

programa2.c, cambiando el gets.txt por put.txt para que lea el programa. esto lo hicimos porque el fputs crea el archivo además de que nos deja escribir sobre el

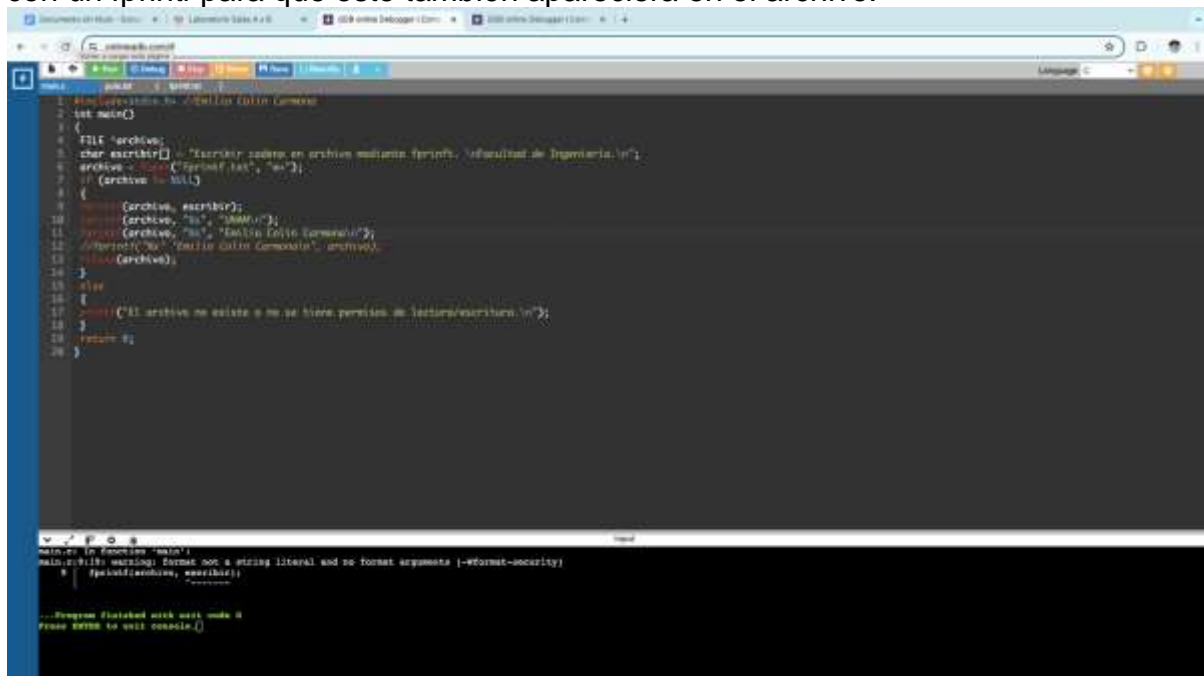


```
1 #include <stdio.h> //Emilio Collin Carmona
2 int main()
3 {
4     FILE *archivo;
5     char caracteres[50];
6     archivo = fopen("puts.txt", "r");
7
8     if (archivo != NULL)
9     {
10        printf("El archivo se abrió correctamente.");
11        printf("\nContenido del archivo:\n");
12        while ((fgetc(archivo) != 0))
13        {
14            fgetc(caracteres, 50, archivo);
15            printf("%s", caracteres);
16        }
17        fclose(archivo);
18    }
19
20    return 0;
21 }
```

El archivo se abrió correctamente.
Contenido del archivo:
Escribir cadena en archivo mediante fputs.
Facultad de Ingeniería.
Facultad de Ingeniería.

...Program finished with exit code 0
Press ENTER to exit console.

3- programa5.c modifique el programa otra vez cambiando el r de read por la w de write y agregue mi nombre con un fprintf para que este también apareciera en el archivo.



```
1 #include <stdio.h> //Emilio Collin Carmona
2 int main()
3 {
4     FILE *archivo;
5     char escribir[] = "Escribir cadena en archivo mediante fprintf. \nFacultad de Ingeniería.\n";
6     archivo = fopen("puts.txt", "w");
7     if (archivo != NULL)
8     {
9         fprintf(archivo, escribir);
10        fputs(archivo, "mi", "Emilio\n");
11        printf(archivo, "mi", "Emilio Collin Carmona");
12        //fprintf("mi", "Emilio Collin Carmona", archivo);
13        fclose(archivo);
14    }
15    else
16    {
17        printf("El archivo no existe o no se tiene permiso de lectura/escritura.\n");
18    }
19    return 0;
20 }
```

main.c: In function 'main':
main.c:11:18: warning: format not a string literal and no format arguments [-Wformat-security]
11 | fputs(archivo, "mi", "Emilio Collin Carmona");
| ~~~~~

...Program finished with exit code 0
Press ENTER to exit console.


```
fp22alu14 -- vi programa6.c -- 112x41
#include <stdio.h>
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\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;
}

"programa6.c" 29L, 592B
```

Después checo si el programa está bien con “gcc programa6.c -o programa6.out”.
Después lo ejecuto con el comando “./programa6.out”.

```
fp22alu14 -- bash -- 112x41
12 | \n\tnombre\tprograma nombre_archivo\n");
    | ^
programa6.c:12:4: error: stray '\' in program
12 | \n\tnombre\tprograma nombre_archivo\n");
    | ^
programa6.c:12:3: error: 'n' undeclared (first use in this function)
12 | \n\tnombre\tprograma nombre_archivo\n");
    | ^
programa6.c:12:3: note: each undeclared identifier is reported only once for each function it appears in
programa6.c:12:4: error: expected ')' before 'nombre_'
12 | \n\tnombre\tprograma nombre_archivo\n");
    | ^~~~~~
programa6.c:12:13: error: stray '\' in program
12 | \n\tnombre\tprograma nombre_archivo\n");
    | ^
programa6.c:12:38: error: stray '\' in program
12 | \n\tnombre\tprograma nombre_archivo\n");
    | ^
programa6.c:12:40: warning: missing terminating " character
12 | \n\tnombre\tprograma nombre_archivo\n");
    | ^~~~~
programa6.c:12:40: error: missing terminating " character
12 | \n\tnombre\tprograma nombre_archivo\n");
    | ^~~~~
programa6.c:13:11: error: expected ';' before '}' token
13 |     return 1;
    |     ^
14 | }
    | ^
Brasil27:~ fp22alu14$ vi programa6.c
Brasil27:~ fp22alu14$ gcc programa6.c -o programa6.out
Brasil27:~ fp22alu14$ vi programa6.c
Brasil27:~ fp22alu14$ vi programa6.c
Brasil27:~ fp22alu14$ ./programa6.c
-bash: ./programa6.c: No such file or directory
Brasil27:~ fp22alu14$ ./programa6.out
Ejecutar el programa de la siguiente manera:
    nombre_programa nombre_archivo
Brasil27:~ fp22alu14$ gcc programa6.c -o programa6.out
```

Utilice el “./programa6.out programa6.c” para poder visualizar el programa6.c en la terminal.

```

fp22alu14 ~ -bash -- T12x81
Brasil127:~ fp22alu14$ gcc programa6.c -o programa6.out
Brasil127:~ fp22alu14$ vi programa6.c
Brasil127:~ fp22alu14$ vi programa6.c
Brasil127:~ fp22alu14$ ./programa6.c
-bash: ./programa6.c: No such file or directory
Brasil127:~ fp22alu14$ ./programa6.out
Ejecutar el programa de la siguiente manera:
    nombre_programa nombre_archivo
Brasil127:~ fp22alu14$ ./programa6.out /Users/fp22alu14/Desktop/fprintf.txt
El archivo /Users/fp22alu14/Desktop/fprintf.txt no existe o no se puede abrirBrasil127:~ fp22alu14$ ./programa6.o
ut programa6.c
#include <stdio.h>
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_programa 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;
}
???Brasil127:~ fp22alu14$

```

Ahora hago lo mismo con el programa7.c.

```

fp22alu14 ~ vi programa7.c -- T12x41
#include <stdio.h>
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;
}
~
~
~
*programa7.c* 37L, 964B

```

Con gcc lo compilo y ahora lo ejecuto con "./", lo copio en "archivo.txt." y uso el contenido del programa anterior para ver el archivo.

```

fp22alu14$ gcc programa7.c -o programa7.out
fp22alu14$ ./programa7.out programa7.c archivo2.txt
fp22alu14$ cat archivo2.txt
#include <stdio.h>
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;
}
fp22alu14$

```

TAREA:

1.- Completa el cuadro referente a funciones, sintaxis, ejemplo y características de cada una de las funciones vistas en el laboratorio.

Función	Sintaxis	Características	Ejemplo de sintaxis
fputs()	fputs(char *buffer, FILE *apArch);	La función fputs() permite escribir una cadena en un archivo específico	fputs (listaNombre, archivo);
fgets()	char *fgets(char *buffer, int tamaño, FILE *apArch);	La función fgets() permite leer una cadena desde el archivo especificado. E	fgets(linea, sizeof(linea), archivo);
fopen()	*FILE fopen(char *nombre_archivo, char *modo);	La función fopen() abre una secuencia para que pueda ser utilizada y la asocia a un archivo.	archivo = fopen("miArchivo.txt", "r");
fclose()	int fclose (FILE *apArch);	La función fclose() cierra una secuencia que fue abierta mediante una llamada a fopen().	fclose(archivo);
fprintf()	int fprintf(FILE *apArch, char *formato, ...);	se comporta similarmente a printf() (imprimir), con la diferencia de que opera sobre un archivo.	fprintf(archivo, "Nombre: %s, Edad: %d", nombre, edad);
fscanf()	int fscanf(FILE *apArch, char *formato, ...);	se comporta similarmente a scanf() (leer), con la diferencia de que opera sobre un archivo.	fscanf(archivo, "%s %d", nombre, &edad);

fread()	int fread(void *ap, size_t tam, size_t nelem, FILE *archivo)	fread permite leer uno o varios elementos de la misma longitud a partir de una dirección de memoria determinada (apuntador).	fread(buffer, sizeof(int), 10, archivo);
fwrite()	int fwrite(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.	fwrite(datos, sizeof(int), 5, archivo);

CONCLUSIÓN:

En esta última practica se cumplió el objetivo ya que a través de todos los ejercicios vistos en clase y un poco en la tarea vimos los diferentes tipos de funciones que podíamos ejecutar y como se mostraban en el archivo. Desde la función fgets y fputs hasta la función fprintf y fscanf que son muy similares a lo que hacen printf y scanf pero estas como todas las demás funciones operaban sobre el archivo. Todas las funciones las vimos en cada actividad que hubo y al trabajar en terminal esta vez no tuve tanta dificultad en realizar la actividad como en practicas anteriores, mostrando que he mejorado al menos al trabajar en terminal, comparando con las primeras prácticas.

Sentí un poco más sencilla esta práctica, probablemente como fue la última del semestre pero aun así siempre pude poner en parte cosas que fui viendo todo el semestre en esta materia y cada nueva practica pude aprender algo nuevo que en esta ocasión, pude realizar sin mayor dificultad.

BIBLIOGRAFÍA:

- Solano, J., García, E., Sandoval, L., Nakayama, A., Arteaga, I. y Castañeda, M., et al. (2022). Manual de prácticas del laboratorio de Fundamentos de programación. Facultad de Ingeniería. Consultado de <http://lcp02.fi-b.unam.mx> el 4 de noviembre de 2024.
- S.a. (2024). Fgets () – leer una serie. IBM. consultado de <https://www.ibm.com/docs/es/i/7.5?topic=functions-fgets-read-string> el 4 de noviembre de 2024.
- S.a. (2024). fputs ()- Serie de escritura. IBM. consultado de <https://www.ibm.com/docs/es/i/7.5?topic=functions-fputs-write-string> el 4 de noviembre de 2024.

LIGA GITHUB: <https://github.com/EmilioCC11/FDLP-Emilio-Colin>