	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	Marco Antonio Martinez Quintana
Asignatura:	Fundamentos de Programación
Grupo:	03
No de Práctica(s):	Practica N°13
Integrante(s):	Torres Gracian Christian Ivan
No. de Equipo de cómputo empleado:	No Aplica
No. de Lista o Brigada:	55
Semestre:	Semestre N°1
Fecha de entrega:	
Observaciones:	

CALIFICACIÓN: _____

Guía práctica de estudio 13: Lectura y escritura de datos

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.

Actividades:

- A través de programas en C, emplear las funciones para crear, leer, escribir y sobrescribir archivos de texto plano.
- Manipular archivos empleando los diferentes tipos de acceso a ellos.

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.

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*.

Licencia GPL de GNU

El software presente en esta práctica es libre bajo la licencia GPL de GNU, es decir, se puede modificar y distribuir mientras se mantenga la licencia GPL.

```
/*
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 *
 * Author: Jorge A. Solano
 *
*/
```

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)

```c
#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;
}
```

Funciones fgets y fputs

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 especifico. 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 lee 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;
}
```

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

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 (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 fprinft. \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

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;
}
```

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 sobreescribe 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;
}
```

Resultados

```c
#include<stdio.h>
int GSP(int n)
{
    //Apuntador al archivo
    FILE *b;
    b=fopen("ResultadosFactorial.txt","w");

    int r=1, i=1;

    while(i<=n)
    {
        fprintf(b,"%d * %d = ",i,r);

        r=r*i;
        i++;

        fprintf(b,"%d \n",r);
    }

    return r;

    fclose(b);
}

int main()
{
    //variables
    char u=163, sp=168, e=130;
    int n,i,res;

    //Bienvenida
    printf("\n\n\t\t\t\t Factorial de un n%cmero\n\n ",u);

    //Solisitar valores
    printf("\n\t %cDe qu%c n%cmero desaeas saber su factorial:  ",sp,e,u);
    scanf("%d",&n);

    //Llamado a la funcion
    res=GSP(n);

    //Resultado
    printf("\n\t %cl factorial del n%cmero dado es: %i",e,u,res);

    return 0;
}
```

```
C:\Users\ivan-\Lenguaje C>gcc "FactArchivo .c" -o "FactArchivo .exe"

C:\Users\ivan-\Lenguaje C>"FactArchivo .exe"


                                Factorial de un número


        ¿De qué número desaeas saber su factorial:  7

        él factorial del número dado es: 5040
C:\Users\ivan-\Lenguaje C>
```

ResultadosFactorial: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
1 * 1 = 1
2 * 1 = 2
3 * 2 = 6
4 * 6 = 24
5 * 24 = 120
6 * 120 = 720
7 * 720 = 5040
```

Bibliografía

El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991.