Práctica 3: Entrada / Salida. Ficheros

.::2 Sesiones::.

0.- Objetivos de la práctica

Los objetivos de esta práctica son: (1) realizar una programación correcta y estructurada; (2) manejar el acceso a ficheros de datos desde un programa en C.

1.- Declaración de un fichero

Sintaxis:

```
FILE *variable_fichero;
```

ej: FILE *fich;

2.- Apertura y cierre de ficheros

Los pasos para utilizar un fichero son los siguientes:

1) Debemos realizar una **operación de apertura** del fichero antes de proceder a su uso

2) Con el fichero podemos realizar dos operaciones:

   - Para almacenar datos debemos realizar una *operación de escritura*
   - Para obtener datos de él, debemos realizar una *operación de lectura*

3) Cuando hayamos finalizado con ese fichero, debemos realizar una **operación de cierre,** con el fin de liberar memoria.

*2.1.- Operación de apertura: función **fopen()***

Para abrir un fichero debemos usar la función fopen:

Ej: fich = fopen(nombre,modo);

La dirección que devuelve la función fopen() se le asigna a la variable de tipo fichero que lo identificará.

Los parámetros que se le pasan a la función serán:

- **nombre** → que corresponde al nombre del fichero, correspondiente a una cadena de caracteres que contenga el nombre y ruta para acceder a él.
- **modo** → que corresponde a una cadena de caracteres que indica el tipo del fichero (texto o binario) y que tipo de uso se le va a dar

Modos de apertura de ficheros	
r	Lectura
w	Escritura. Si no existe el fichero lo crea
r+	Lectura / Escritura
w+	Lectura / Escritura. Si no existe el fichero lo crea
a	Añadir al final
a+	Lectura / Añadir al final. Si no existe lo crea

Ejemplo:

```
fich = fopen("alumnos.txt","r+");
```

En este ejemplo vemos que indicamos el modo de *Lectura/Escritura* al fichero *alumnos.txt*

Si ocurriera algún error al intentar abrir el fichero, el valor que devolvería la función fopen sería NULL.

Altenativamente, podemos pasarle una **ruta relativa** a la función `fopen`:

```
fich = fopen("..\\directorio\\alumnos.txt", "w");
```

2.2.- Lectura/Escritura sobre un fichero

Una vez que hemos abierto el fichero, procederemos a leer o escribir de él o sobre él.
Entre las funciones para leer o escribir datos tenemos:

fscanf-fprintf, fgetc-fputc, fgets-fputs

Se recomienda utilizarlas por parejas, tal y como se indica.

- Leer y escribir con formato en archivos de texto → fscanf-fprintf
  Los prototipos de las funciones son:

```
int fscanf(FILE *fp,"cadena de control",otros_argumentos);
int fprintf(FILE *fp, "cadena de control",otros_argumentos);
```

Ejemplo:

```
#include <stdio.h>
main()
{
    int x,y=3;
    FILE *f,*g;
    f=fopen("holaf.txt","w");
    g=fopen("holag.txt","r+");
    fscanf(fichero,"%d",&x);
    fprintf(fichero,"%d",y);
    system("PAUSE");
}
```

- Leer y escribir caracteres → fgetc-fputc
  Los prototipos de las funciones son:

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

```
#include <stdio.h>
main()
{
    char c;
    FILE *fich;
    fputc(c,fich);
    fgetc(fich);
}
```

- Leer y escribir cadenas de caracteres → fgets-fputs
  Los prototipos de las funciones son:

```
char * fgets ( char * str, int num, FILE * stream );
int fputs ( const char * str, FILE * stream );
```

-fgets() funciona de la siguiente manera:

```
#include <stdio.h>

int main()
{
   char nombre[10]="datos.dat", linea[81];
   FILE *fichero;

   fichero = fopen( nombre, "r" );
   printf( "Fichero: %s -> ", nombre );
   if( fichero )
      printf( "existe (ABIERTO)\n" );
   else
   {
      printf( "Error (NO ABIERTO)\n" );
      return 1;
```

```
   }

   printf( "La primera linea del fichero: %s\n\n", nombre );
   printf( "%s\n", fgets(linea, 81, fichero) );

   if( !fclose(fichero) )
      printf( "\nFichero cerrado\n" );
   else
   {
      printf( "\nError: fichero NO CERRADO\n" );
      return 1;
   }

   return 0;
}
```

Como vemos en el ejemplo, en fgets() hemos puesto primero la variable, a continuación le hemos dicho que solo capture los ochenta y un primeros dígitos (con esto evitamos que si escribimos una oración de cien letras no haya memoria y se desborde el programa)

-fputs() funciona de la siguiente manera:

```
fputs( "Esto es un ejemplo usando \'fputs\'\n", fichero );
```

2.3.- Marca de final de fichero

En cualquier archivo tendremos una marca de final de fichero que nos indica que a partir de ella no hay más datos. Una posible utilidad de esta función es leer el contenido del fichero mientras no encontremos la marca EOF.

2.4.- Operación para el cierre de ficheros

Siempre que acabemos de trabajar con el fichero, debemos de realizar un operación de cierre del mismo. La función que se encarga de ello es `fclose(FILE *fp)`

Ejemplo:

```
fclose(fp);
```

3.- Ejercicios a entregar:

Ejercicio 1

Realizar un programa en C que sea capaz de leer un fichero ya existente que únicamente contiene un número entero. Para ello, crear con el "Bloc de Notas" de Windows un fichero llamado "ejercicio1.txt". Mostrar ese número entero por pantalla*.*

Utilizar la función fscanf() para realizar la lectura del fichero.

Ejercicio 2

Realizar un programa en C que sea capaz de leer un fichero llamado "ejercicio2.txt" ya existente que contiene un número entero seguido de una cadena de caracteres, tal y como se muestra a continuación:

2 cadena

Utilizar la función fscanf() para realizar la lectura del fichero.

Mostrar el contenido del fichero por pantalla

Ejercicio 3

Realizar un programa en C capaz de escribir en un fichero que no existe previamente, y al que llamaremos ejercicio3.txt, el número entero "6" y la cadena de caracteres "hola" quedando de esta forma:

6 hola

Utilizar la función fprintf() para realizar la escritura en el fichero.

Ejercicio 4

Teniendo previamente un archivo llamado "nuevo.txt", que contiene únicamente la cadena "holaMundo", realizar un programa en C que abra dicho archivo en modo lectura y mientras no encontremos la marca EOF leemos carácter a carácter del fichero y lo imprimimos por pantalla.

Utilizar la función feof() para encontrar la marca de final de fichero y utilizar las funciones putchar () y fgetc ()

Ejercicio 5

Realizar un programa en C que sea capaz de leer del fichero "ejercicio5.txt" ya existente, y que contiene la palabra "linux". Debe extraer de él el primer carácter. Además, debe mostrar ese carácter por pantalla*.*

Utilizar la función `fgetc() para extraer dicho carácter.`

Ejercicio 6

Realizar un programa en C que introduzca las 26 primeras letras del abecedario inglés en un fichero llamado "ejercicio6.txt", el cual crearemos desde la propia función main con la apertura correspondiente para escritura.

Ejercicio 7

Se pide al alumno que escriba el código siguiente y que compruebe su funcionamiento.

```
    char nombre[LONG_CADENA];
    char nombreLargo[LONG_CADENA];
    int numero=100;
    char carac = 'A';
    FILE *fp;

1:  printf("Intro nombre sin extension");
2:  scanf("%s", nombre);

3:  strcpy(nombreLargo, "..\\out\\");
4:  strcat(nombreLargo, nombre);
5:  strcat(nombreLargo, ".txt");

    printf("\nAbriendo fichero %s", nombreLargo);

6:  fp = fopen(nombreLargo, "w");
    if(fp==NULL){
        printf("\nError abriendo fichero");
         return;
      }

7:  fprintf(fp, "%d\n", numero);
8:  fprintf(fp, "%c\n", numero);

9:  fprintf(fp, "%d\n", carac);
10: fprintf(fp, "%c\n", carac);

11: fclose(fp);
```

El programa recoge el nombre del fichero por teclado (sin extensión, líneas 1, 2). A continuación añade la ruta relativa donde se encuentra el fichero, así como la extensión (líneas 3 a 5). Por ejemplo, si el usuario teclea prueba, forma la cadena: ..\out\prueba.txt. En la línea 6 se abre efectivamente el fichero. En las líneas 7-10 se escribe información en el fichero.

El alumno deberá ejecutar el programa anterior y observar el fichero de salida generado (p.e. usando el programa 'Bloc de notas' de Windows). ¿Cómo se explican los datos del fichero?

Ejercicio 8

El alumno deberá escribir el "ejercicio 8.c" siguiendo las instrucciones que se dan a continuación.

El programa deberá realizar las siguientes funciones:

a) Leer cadenas de texto por teclado grabándolas en un fichero con extensión **.txt**. Se guardará cada cadena en una línea. El nombre del fichero donde se guardarán las cadenas también se introducirá por teclado (sin extensión). El programa leerá cadenas hasta que se teclee "**salir**", en mayúscula o minúscula.
b) Leer todas las cadenas del fichero anterior imprimiendo en pantalla cada cadena junto con su longitud y generando un nuevo fichero en el que se guardará cada cadena junto con su longitud en una línea. Este fichero se creará también con extensión **.txt**, y su nombre se introducirá por teclado.
c) Concatenar los dos ficheros generados en la práctica creando uno nuevo con extensión **.txt**. El nombre de este tercer fichero se introducirá por teclado.
d) Debe indicarse si se ha producido algún error en el proceso.

A continuación se muestra un **ejemplo** de ejecución del programa (en negrita aparecen los datos introducidos por el usuario):

```
Introducir el nombre del fichero sin extension:
fichero1

Introducir las cadenas (salir para terminar):
a
ab
abc
abcd
salir

Introducir el nombre del fichero donde incluir las longitudes:
fichero2

Cadenas junto con longitudes:
a:1
ab:2
abc:3
abcd:4

Introducir el nombre del fichero concatenado:
fichero3

Programa finalizado con exito.
```

Se deberán seguir los siguientes pasos:

1) El fichero fuente se llamará *ejercicio8.c*.

2) Definir la función ***int GenerarFichero(char *nombrefich)***, que debe:

   a) Abrir el fichero cuyo nombre viene indicado en el parámetro (nombrefich).
   b) Leer cadenas de texto por el teclado (se recomienda utilizar la función ***gets***) grabándolas en el fichero (se recomienda utilizar la función ***fprintf***). Debe existir un comando para finalizar ("**salir**").
   c) Cerrar el fichero.
   d) La función devolverá 0 si no se ha producido ningún error, y -1 en caso contrario.

3) Definir la función ***int CrearFicheroLongitudes(char *fich1, char *fich2)***, que debe

   a) Abrir el fichero ***fich1*** para lectura y el fichero ***fich2*** para escritura.
   b) Leer las cadenas que contenga el primer fichero (utilizar la función ***feof*** para detectar cuando se ha llegado al final del fichero).
   c) Para cada cadena leída (se recomienda hacerlo con ***fgets*** para poder leer cadenas con espacios) del primer fichero se calculará su longitud y se mostrará en pantalla la cadena junto con su longitud. Así mismo se escribirá en el segundo fichero cada cadena junto con su longitud en una línea. Se debe actuar con precaución con la función ***fgets*** comprobando en todo momento si se ha producido un error de lectura antes de realizar cualquier operación. Para calcular la longitud de una cadena se recomienda utilizar la función ***strlen***.
   d) Esta función devolverá 0 si no se ha producido ningún error, y –1 en caso contrario.

4) Para concatenar dos ficheros se recomienda implementar la función ***int ConcatenarFicheros(char *nombrefich1, char *nombrefich2, char *nombrefich3)*** donde los dos primeros parámetros es el nombre de los ficheros a concatenar (deben abrirse en *modo lectura*) y el último es el fichero resultado (abrir en *modo escritura*). El esquema de esta función es como sigue:

   a) Se irán leyendo líneas del primer fichero (se recomienda utilizar la función ***fgets***), escribiéndolas en el fichero resultado (se recomienda utilizar la función ***fputs***). Como se indicó anteriormente, se debe actuar con precaución con la función ***fgets*** comprobando en todo momento si se ha producido un error de lectura antes de realizar cualquier operación.
   b) Posteriormente se leerán líneas desde el segundo fichero (***fgets***) y se escribirán en el fichero resultado (***fputs***), a continuación de las cadenas escritas procedentes del primer fichero.
   c) Se cerrarán todos los ficheros.

   Si se ha producido algún error en esta función, devolver el valor –1. En caso contrario, devolver un 0.

5) Definir la función principal del programa (***main***).

Notas:

- El programa debe estar estructurado dividiendo cada tarea en funciones.
- *Se recomienda diseñar y comprobar cada función independientemente del resto del programa.*
- Se considerará que el tamaño máximo de una cadena introducida por el usuario es de 100 caracteres.
- Utilizar las funciones descritas en la librería estándar de Entrada /Salida (cabecera **<stdio.h>**) descritas en las transparencias para el manejo de ficheros.
- Para el manejo de cadenas puede utilizarse la librería de funciones *string* (cabecera **<string.h>**). A continuación se resumen las funciones más comunes (utilizar la ayuda de Visual C++ para obtener más información).

Prototipo de la función	Descripción
size_t strlen(const char *string);	Devuelve la longitud de una cadena
char *strcat(char *strDestination, const char *strSource);	Añade la cadena strSource al final de la cadena strDestination
int strncmp(const char *string1, const char *string2, size_t count);	Compara los *count* primeros caracteres de dos cadenas (0 SI SON IGUALES)
int strcmp(const char *string1, const char *string2);	Compara dos cadenas (0 SI SON IGUALES)
char *strcpy(char *strDestination, const char *strSource);	Copia el contenido de una cadena en otra