LENGUAJE C

Tema 8 – Ficheros

FICHEROS EN C

- Los ficheros, en contraposición con las estructuras de datos vistas hasta ahora (variables simples, vectores, registros, etc.), son estructuras de datos almacenadas en memoria secundaria.
- El formato de declaración de un fichero es el siguiente:

```
FILE * nom_ var_fich;
```

- En C todos los ficheros almacenan bytes y es cuando se realiza la apertura y la escritura cuando se decide cómo y qué se almacena en el mismo; durante la declaración del fichero no se hace ninguna distinción sobre el tipo del mismo.
- En la operación de apertura se puede decidir si el fichero va a ser de texto o binario, los primeros sirven para almacenar caracteres, los segundos para almacenar cualquier tipo de dato.

APERTURA Y CIERRE DE FICHEROS

- La instrucción más habitual para abrir un fichero es :

```
FILE * fichero;
fichero = fopen ( nombre-fichero, modo);
```

- La función fopen devuelve un puntero a un fichero que se asigna a una variable de tipo fichero. Si existe algún tipo de error al realizar la operación, por ejemplo, porque se desee abrir para leerlo y éste no exista, devuelve el valor **NULL**.
- El nombre-fichero será una cadena de caracteres que contenga el nombre (y en su caso la ruta de acceso) del fichero tal y como aparece para el sistema operativo.
- El modo es una cadena de caracteres que indica el tipo del fichero (texto o binario) y el uso que se va ha hacer de él lectura, escritura, añadir datos al final, etc.

  Cuando se termine el tratamiento del fichero hay que cerrarlo; si la apertura se hizo con fopen el cierre se hará con `fclose (nombre-fichero);`
- Para utilizar las instrucciones de manejo de ficheros que veremos en esta unidad es necesario incluir la librería <stdio.h>.

MODOS DE APERTURA

Los modos disponibles son:

- r abre un fichero para lectura. Si el fichero no existe devuelve error.
- w abre un fichero para escritura. Si el fichero no existe se crea, si el fichero existe se destruye y se crea uno nuevo.
- a abre un fichero para añadir datos al final del mismo. Si no existe se crea.
- + símbolo utilizado para abrir el fichero para lectura y escritura.
- b el fichero es de tipo binario.
- t el fichero es de tipo texto.

Si no se pone ni b ni t el fichero es de texto. Los modos anteriores se combinan para conseguir abrir el fichero en el modo adecuado.

Por ejemplo, para abrir un fichero binario ya existente para lectura y escritura el modo será "rb+ "; si el fichero no existe, o aun existiendo se desea crear, el modo será " wb+ ". Si deseamos añadir datos al final de un fichero de texto bastará con poner "a", etc.

LECTURA Y ESCRITURA DE FICHEROS

Para almacenar datos en un fichero es necesario realizar una operación de escritura, de igual forma que para obtener datos hay que efectuar una operación de lectura. En C existen muchas y variadas operaciones para leer y escribir en un fichero; entre ellas tenemos:

`fread-fwrite, fgetc-fputc, fgets-fputs, fscanf-fprintf`

Es aconsejable utilizarlas por parejas; es decir, si se escribe con `fwrite` se debe leer con `fread`.

FREAD - FWRITE

Función fread:

```
fread(void *puntero, size_t tamaño, size_t nregistros, FILE *fichero);
```

Esta función está pensada para trabajar con registros de longitud constante. Es capaz de leer desde un fichero uno o varios registros de la misma longitud y a partir de una dirección de memoria determinada. El usuario es responsable de asegurarse de que hay espacio suficiente para contener la información leída.
El valor de retorno es el **número de registros leídos**, no el número de bytes. Los parámetros son: un puntero a la zona de memoria donde se almacenarán los datos leídos, el tamaño de cada registro, el número de registros a leer y un puntero a la estructura FILE del fichero del que se hará la lectura.

Función fwrite:

```
fwrite(void *puntero, size_t tamaño, size_t nregistros, FILE *fichero);
```

Esta función también está pensada para trabajar con registros de longitud constante y forma pareja con fread. Es capaz de escribir hacia un fichero uno o varios registros de la misma longitud almacenados a partir de una dirección de memoria determinada.
El valor de retorno es el **número de registros escritos**, no el número de bytes. Los parámetros son: un puntero a la zona de memoria donde se almacenarán los datos leídos, el tamaño de cada registro, el número de registros a leer y un puntero a la estructura FILE del fichero del que se hará la lectura.

FPRINTF - FSCANF

Función fprintf:

```
fprintf(FILE *fichero, const char *formato, ...);
```

La función fprintf funciona igual que printf en cuanto a parámetros, pero la salida se dirige a un fichero en lugar de a la pantalla.

Función fscanf:

```
fscanf(FILE *fichero, const char *formato, ...);
```

La función fscanf funciona igual que scanf en cuanto a parámetros, pero la entrada se toma de un fichero en lugar del teclado.

Función fflush:

```
fflush(FILE *fichero);
```

Esta función fuerza la salida de los datos acumulados en el buffer de salida del *fichero*. Para mejorar las prestaciones del manejo de ficheros se utilizan buffers, almacenes temporales de datos en memoria, las operaciones de salida se hacen a través del buffer, y sólo cuando el buffer se llena se realiza la escritura en el disco y se vacía el buffer. En ocasiones nos hace falta vaciar ese buffer de un modo manual, para eso sirve ésta función.

FGETC- FPUTC

Función fgetc:

```
fgetc(FILE *fichero);
```

Esta función lee un carácter desde un fichero.
El valor de retorno es el carácter leído como un **unsigned char** convertido a **int**. Si no hay ningún carácter disponible, el valor de retorno es EOF. El parámetro es un puntero a una estructura FILE del fichero del que se hará la lectura.

Función fputc:

```
fputc(int caracter, FILE *fichero);
```

Esta función escribe un carácter a un fichero.
El valor de retorno es el carácter escrito, si la operación fue completada con éxito, en caso contrario será EOF. Los parámetros de entrada son el carácter a escribir, convertido a **int** y un puntero a una estructura FILE del fichero en el que se hará la escritura.

FGETS - FPUTS

Función fgets:

```
fgets(char *cadena, int n, FILE *fichero);
```

Esta función está diseñada para leer cadenas de caracteres. Leerá hasta n-1 caracteres o hasta que lea un retorno de línea. En este último caso, el carácter de retorno de línea también es leído.
El parámetro *n* nos permite limitar la lectura para evitar desbordar el espacio disponible en la *cadena*.
El valor de retorno es un puntero a la cadena leída, si se leyó con éxito, y es NULL si se detecta el final del fichero o si hay un error. Los parámetros son: la cadena a leer, el número de caracteres máximo a leer y un puntero a una estructura FILE del fichero del que se leerá.

Función fputs:

```
fputs(const char *cadena, FILE *stream);
```

La función fputs escribe una cadena en un fichero. No se añade el carácter de retorno de línea ni el carácter nulo final.
El valor de retorno es un número no negativo o EOF en caso de error. Los parámetros de entrada son la cadena a escribir y un puntero a la estructura FILE del fichero donde se realizará la escritura.

FUNCIONES DE FICHEROS

Función feof:

```
feof(FILE *fichero);
```

Esta función sirve para comprobar si se ha alcanzado el final del fichero. Muy frecuentemente deberemos trabajar con todos los valores almacenados en un archivo de forma secuencial, la forma que suelen tener los bucles para leer todos los datos de un archivo es permanecer leyendo mientras no se detecte el fin de fichero. Esta función suele usarse como prueba para verificar si se ha alcanzado o no ese punto.
El valor de retorno es distinto de cero sólo si no se ha alcanzado el fin de fichero. El parámetro es un puntero a la estructura FILE del fichero que queremos verificar.

Función rewind:

```
rewind(FILE *fichero);
```

Es una función heredada de los tiempos de las cintas magnéticas. Literalmente significa "rebobinar", y hace referencia a que para volver al principio de un archivo almacenado en cinta, había que rebobinarla. Eso es lo que hace ésta función, sitúa el cursor de lectura/escritura al principio del archivo.
El parámetro es un puntero a la estructura FILE del fichero que queremos rebobinar.

FUNCIONES DE FICHEROS 2

Función fseek:

```
fseek(FILE *fichero, long int desplazamiento, int origen);
```

Esta función sirve para situar el cursor del fichero para leer o escribir en el lugar deseado.
El valor de retorno es cero si la función tuvo éxito, y un valor distinto de cero si hubo algún error.
Los parámetros de entrada son: un puntero a una estructura FILE del fichero en el que queremos cambiar el cursor de lectura/escritura, el valor del desplazamiento y el punto de origen desde el que se calculará el desplazamiento.
El parámetro *origen* puede tener tres posibles valores:
`SEEK_SET` el desplazamiento se cuenta desde el principio del fichero. El primer byte del fichero tiene un desplazamiento cero.
`SEEK_CUR` el desplazamiento se cuenta desde la posición actual del cursor.
`SEEK_END` el desplazamiento se cuenta desde el final del fichero.

Función ftell:

```
ftell(FILE *fichero);
```

La función ftell sirve para averiguar la posición actual del cursor de lectura/escritura de un fichero.
El valor de retorno será esa posición, o -1 si hay algún error.
El parámetro de entrada es un puntero a una estructura FILE del fichero del que queremos leer la posición del cursor de lectura/escritura.

BIBLIOGRAFÍA & LICENCIA

- Textos tomados, corregidos y modificados de diferentes páginas de Internet, tutoriales y documentos, entre los que destaco el libro: *C/C++ Curso de programación*, 2da Ed, Javier Ceballos, Alfaomega Ra-Ma.
- Este documento se encuentra bajo Licencia Creative Commons Attribution – NonCommercial - ShareAlike 4.0 International (CC BY-NC-SA 4.0), por la cual se permite su exhibición, distribución, copia y posibilita hacer obras derivadas a partir de la misma, siempre y cuando se cite la autoría del Prof. Matías E. García y sólo podrá distribuir la obra derivada resultante bajo una licencia idéntica a ésta.
- Autor:

Matías E. García

Prof. & Tec. en Informática Aplicada

www.profmatiasgarcia.com.ar

info@profmatiasgarcia.com.ar