La estructura FILE es un tipo de objeto capaz de almacenar toda la información necesitada para controlar un stream, incluyendo su indicador de posición de fichero, un puntero a su almacenamiento asociado (si existe), un indicador de errores que registra si se ha producido un error de lectura y/o ecsritura, y un indicador de final de fichero que registra si se ha llegado al final del fichero.

La definición de esta estructura depende del compilador; no obstante, suele tener todas las características anteriormente mencionadas. Algunas de sus definiciones son:

1. Compilador DJGPP, versión de GCC para sistemas operativos MS-DOS:

typedef struct {

int \_cnt;

char \* \_ptr;

char \* \_base;

int \_bufsiz;

int \_flag;

int \_file;

char \* \_name\_to\_remove;

int \_fillsize;

} FILE;

Cada uno de los campos de esta definición son:

* \_cnt: número de caracteres restantes en el buffer.
* \_ptr: puntero a la posición actual en el buffer.
* \_base: puntero al inicio del buffer.
* \_bufsiz: tamaño del buffer.
* \_flag: indicadores (si está en lectura, escritura, error, EOF, etc.).
* \_file: descriptor de archivo (número asignado por el SO).
* \_name\_to\_remove: nombre del archivo temporal a eliminar al cerrar.
* \_fillsize: tamaño de llenado del buffer.

1. Compilador Borland C++ / Turbo C para sistemas operativos MS-DOS y Windows 16-bit:

typedef struct {

int level;

unsigned flags;

char fd;

unsigned char hold;

int bsize;

unsigned char \_FAR \*buffer;

unsigned char \_FAR \*curp;

unsigned istemp;

short token;

} FILE;

Cada uno de los campos de esta definición son:

* level: nivel de buffering.
* flags: indicadores del archivo (modo de apertura, error, etc.).
* fd: file Descriptor (número del archivo en el sistema, solo 1 byte, máximo 256 archivos abiertos).
* hold: último byte leído/escrito (optimización).
* bsize: tamaño del buffer.
* \_FAR \*buffer: puntero a buffer en memoria segmentada (16 bits).
* \_FAR \*curp: puntero a la posición actual en el buffer.
* istemp : si el archivo es temporal.
* token: uso interno del compilador.

1. Compilador Dev-C++, que usa MinGW (Minimalist GNU para Windows) o MSVC(Microsoft Visual C++) para sistemas operativos Windows 32-bit y 64-bit:

typedef struct \_iobuf

{

char\* \_ptr;

int \_cnt;

char\* \_base;

int \_flag;

int \_file;

int \_charbuf;

int \_bufsiz;

char\* \_tmpfname;

} FILE;

Cada uno de los campos de esta definición son:

* \_ptr: puntero a la posición actual en el buffer.
* \_cnt: número de bytes restantes en el buffer.
* \_base: puntero al inicio del buffer.
* \_flag: indicadores (lectura/escritura/error/EOF).
* \_file: descriptor de archivo (soporta más de 256 archivos, porque es int).
* \_charbuf: usado para manejar caracteres individuales en buffering.
* \_bufsiz : tamaño del buffer.
* \_tmpfname: nombre del archivo si es un archivo temporal.

## Indicador de posición

El indicador de posición de un fichero es un puntero interno que se emplea para rastrear la ubicación actual dentro de un archivo; es decir, su función es determinar donde ocurrirán las siguientes operaciones de lectura o escritura.

El valor que almacena esta posición depende en función de si estamos en un fichero binario o en un archivo de texto:

1. Fichero binario: en los fichero binarios los datos se almacenan sin conversión, por lo cual, la posición del indicador siempre coincide con el número de bytes desde el inicio hasta la posición de este. Además, se debe tener en cuenta que en estos ficheros no hay conversión de caracteres, por lo tanto, caracteres como: ‘\n’, se almacenan tal cual sin modificaciones y se tendrán en cuenta a la hora de contar el número de bytes.
2. Fichero de texto: en los ficheros de texto la posición del indicador representa con el número de caracteres desde el inicio hasta la posición de este. No obstante, en función de la conversión de caracteres que realice cada sistema operativo, esto afectará al número de caracteres que se cuente:

* En Unix/Linux, no existe conversión de caracteres, por lo cual, el número de caracteres que se cuente será la cantidad exacta de caracteres que hay en el fichero hasta el inicio de este.
* En Windows, el sistema convierte saltos de línea automáticamente, de manera que el carácter: ‘\n’, en este sistema operativo serían dos caracteres en uno debido a la conversión que hace: ‘\n’ (1 carácter) => ‘\r\n’ (2 caracteres), por lo tanto, al contar la cantidad de caracteres esto alterará la cuenta y no reflejará el número exacto de caracteres desde el inicio (habría que contar que por cada carácter: ‘\n’ que haya en el texto aumentará en una unidad la cuenta).

Cuando se abre un fichero, se asocia una estructura FILE al fichero abierto la cual dentro de uno de sus campos almacenará el indicador de posición de este fichero que apuntará a una ubicación específica dentro del archivo. La posición del fichero donde se localiza inicialmente es:

1. Si el fichero se abre en modo lectura: ‘r’, el indicador de posición apunta al inicio del archivo.
2. Si el fichero se abre en modo escritura: ‘w’, el fichero se trunca y el indicador se coloca al inicio del fichero.
3. Si el fichero se abre en modo adición: ‘a’, el indicador se coloca al final del archivo.

### Función FTELL

La función: ftell, obtiene el valor actual del indicador de posición del fichero para el fichero específico al que apunta.

El prototipo de esta función es:

*long int ftell (FILE\* stream)*

El parámetro que se le deberá pasar a esta función se corresponde con la estructura FILE asociada al fichero del que queremos obtener el indicador de posición.

Esta función devolverá el valor del indicador de posición del fichero si el proceso de obtención tiene éxito. Si existe un error, la función devolverá el valor -1L.

### Función FSEEK

La función: fseek, modifica la posición del indicador de posición del fichero para el fichero específico al que apunta.

El prototipo de esta función es:

*int fseek(FILE\* stream, long int offset, int origin)*

Los parámetros que se deben pasar a esta función son:

* *stream*: puntero a la estructura FILE asociada al fichero en el que se quiere modificar el indicador de posición.
* *offset*: valor del desplazamiento, en bytes, que se aplicará desde el punto de referencia que se indique en el parámetro: *origin*.
* *origin*: valor del punto de referencia desde el cual se aplicará el desplazamiento de la posición del indicador. Este parámetro puede tomar los siguientes valores:
* *SEEK\_SET* (posee el valor 0): establece la posición del fichero desde el inicio del fichero. Por lo tanto, el desplazamiento se aplicará desde el inicio del fichero.
* *SEEK\_CUR* (posee el valor 1): establece la posición del fichero desde la posición actual. Por lo tanto, el desplazamiento se aplicará desde la posición donde se encuentre el indicador de posición del fichero.
* *SEEK\_END* (posee el valor 2): establece la posición del archivo desde el final del fichero. Por lo tanto, el desplazamiento se aplicará desde el final del fichero.

Esta función devolverá un valor distinto a 0 si la modificación de la posición del fichero no se pudo llevar a cabo o el valor 0 si el proceso fue exitoso.

### Función REWIND

La función: rewind, establece el indicador de posición del fichero al inicio del fichero.

Su prototipo de función es:

*void rewind (FILE\* stream)*

El parámetro que se le pasa a esta función es el puntero a la estructura FILE asociada al fichero en el que se va a establecer el indicador de posición al inicio de este.