Archivos

**Técnico Universitario en Programación - Laboratorio de Computación II**

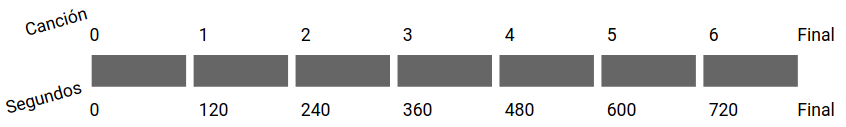
**Simón, Angel**

Con lo visto hasta el momento hemos logrado registrar la información que cargamos en nuestro sistema en un archivo de datos. También, pudimos realizar tareas como consultar si registros existe o no y eventualmente listarlo. O bien, listar todo los registros del archivo.

Nos restaría realizar una tarea más compleja como lo es la modificación de un registro y también conocer la cantidad de registros que contiene nuestro archivo para, por ejemplo, pedir memoria dinámica.

Para realizar este tipo de tareas necesitaremos hacer uso de dos funciones de archivos: fseek y ftell.

La función fseek nos permite mover el indicador de nuestro archivo a la posición (en bytes) que deseemos. Esta funcionalidad es similar al de un cassette de audio o de video.





Supongamos que tenemos un cassette de audio de una banda muy particular cuyos temas tienen todos una duración de 120 segundos. Entonces, podemos deducir que el primer tema arranca en el segundo 0, el próximo tema en el segundo 120, el tercero en el 240 y así sucesivamente.

El reproductor de cassettes no puede ir fácilmente al tema 4 (como sí podría hacerlo con un CD) pero puede ubicarse en el segundo 360 sin mayores dificultades. Será cuestión nuestra transformar ese 360 en nuestro número de canción para poder reproducirla.

Luego, si cada canción tiene una duración de 120 segundos. Podemos llegar a la siguiente conclusión:

| Duración del cassette: 840 segundos  Duración de un track : 120 segundos  Cantidad de tracks : 840 / 120 → 7 tracks |
| --- |

La inversa sería igual de sencillo, si quiero conocer el tiempo inicial donde comienza un track es cuestión de realizar el producto entre el número de track y la duración de cada canción.

| Duración de un track: 120 segundos  # de track a conocer: 6  Tiempo inicial: 120 \* (6-1) → 600 |
| --- |

La misma lógica se puede utilizar para trabajar con archivos. Ya que los registros son de longitud fija. Quien nos permitirá movernos en bytes hacia adelante o atrás en nuestro archivo es la función fseek.

## fseek

| int fseek(FILE \*archivo, long int desplazamiento, int origen); |
| --- |

Mueve una cantidad de bytes indicado por *desplazamiento* desde el *origen* en el puntero FILE indicado por *archivo.*

El origen desde donde se realizará el desplazamiento se describe en la siguiente tabla:

| **Valor** | **Alias** | **Descripción** |
| --- | --- | --- |
| 0 | SEEK\_SET | Inicio del archivo |
| 1 | SEEK\_CUR | Posición actual del archivo |
| 2 | SEEK\_END | Final del archivo |

**Tabla 4 ·** Valores de *origen* para la función fseek.

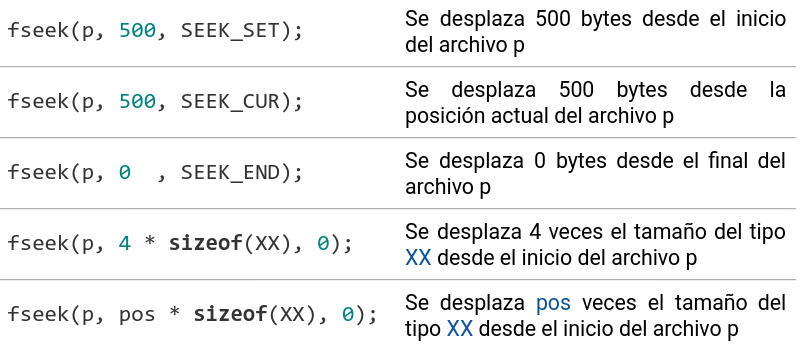
Volvamos a nuestro archivo de Países del ejemplo anterior. Sabemos que cada registro tiene un sizeof o tamaño en bytes de 120. Supongamos que queremos ubicarnos al principio del registro número 4 para poder leerlo. Podemos notar como, al igual que con los vectores, tenemos una indexación en base cero. Es decir, el primer registro se encuentra en el byte 0 y no en el byte 120. Entonces, si queremos leer el cuarto registro entonces la posición contando desde 0 es la número 3.



El código en C++ para leer el cuarto registro de país es la siguiente:

| #include <iostream> using namespace std; #include <cstdio> #include "Pais.h"  int main(){  Pais reg;  FILE \*p = fopen("paises.dat", "rb");  if (p == NULL){  cout << "No se pudo abrir el archivo.";  return 1;  }  int pos = 1;  fseek(p, sizeof(Pais) \* pos, 0);  fread(&reg, sizeof(Pais), 1, p);  fclose(p);  reg.mostrar();  return 0; } |
| --- |

**Código 5 ·** Programa que lee el segundo registro del archivo de países y lo muestra.



**Imagen 2 ·** Otros ejemplos de fseek

Si bien el código es funcional, lo único que variaría si en lugar de querer leer el segundo registro quisiéramos leer el décimo es el valor de *pos*. Todo este conjunto de instrucciones es candidato a ser reemplazados por una función global o por un método de una clase que podríamos llamar PaisArchivo.

Pensemos entonces en un método de la clase PaisArchivo que nos permita leer un registro y guardarlo en un objeto de la clase Pais. Será necesario que reciba la posición en el archivo a leer (indexado en base 0) y que devuelva el objeto leído.

| Pais PaisArchivo::leer(int pos){  Pais reg;  FILE \*p;  p = fopen("paises.dat", "rb");  if (p == NULL){  return reg;  }  fseek(p, pos \* sizeof(Pais), 0);  fread(&reg, sizeof(Pais), 1, p);  fclose(p);  return reg; } |
| --- |

De esta manera, si queremos leer un registro que no existe. Digamos que queremos leer la posición 1000. En lugar de generar un error por pantalla, el método nos devolverá un registro de pais con basura o con lo que el constructor haya establecido. Será cuestión de evaluar el valor devuelto para poder continuar o no con el normal flujo de nuestro programa. Aquí un ejemplo de su uso:

| #include <iostream> using namespace std; #include "Pais.h"  int main(){  PaisArchivo arch;  Pais p = arch.leer(15);  p.mostrar();  return 0; } |
| --- |

**Código 7 ·** Uso del método leer de la clase PaisArchivo.

La salida por pantalla sería la siguiente:

| AUT AT Austria Europa 83859 8091800 1918 77.7 1523 |
| --- |

## ftell

La función ftell es muy sencilla y tiene el propósito de indicarnos cuántos bytes hay entre el comienzo del archivo y el actual indicador del mismo.

Por ejemplo, si el indicador del archivo se encuentra al inicio. La función ftell nos devolverá un cero. Y si se encuentra al final del archivo la función ftell nos devolverá el tamaño en bytes del archivo.

Conocer este dato es muy importante. Ya que saber cuántos bytes tiene en total un archivo es el paso inicial para poder conocer cuántos registros tiene (ver el ejemplo del cassette y la cantidad de tracks).

Veamos esta función:

| int cantidadPaises(){  int bytes;  FILE \*p;  p = fopen("paises.dat", "rb");  if (p == NULL){  return 0;  }  fseek(p, 0, SEEK\_END);  bytes = ftell(p);  fclose(p);  return bytes / sizeof(Pais); } |
| --- |

**Código 8 ·** Función global que devuelve la cantidad de registros del archivo de países.

Si mostramos por pantalla el valor devuelto por la función, nos dará la cantidad de registros que se encuentran en paises.dat, lo cual tiene una utilidad sumamente importante. En primer lugar, es un dato fundamental si lo que queremos es pedir memoria dinámica para alojar el archivo en memoria. Pero por otro lado, saber la cantidad de registros sumada al método leer visto anteriormente nos permite resolver un listado secuencial del archivo de una manera muy sencilla. Por lo que la función global tendría mucho más sentido si forma parte de la clase PaisArchivo.

| #include <iostream> using namespace std; #include "Pais.h"  int PaisArchivo::getCantidadRegistros(){  int bytes;  FILE \*p;  p = fopen("paises.dat", "rb");  if (p == NULL){  return 0;  }  fseek(p, 0, SEEK\_END);  bytes = ftell(p);  fclose(p);  return bytes / sizeof(Pais); }  int main(){  PaisArchivo arch;  Pais p;  int cant = arch.getCantidadRegistros(), i;  for(i=0; i<cant; i++){  p = arch.leer(i);  p.mostrar();  }  return 0; } |
| --- |

**Código 9 ·** Programa que lista secuencialmente todos los países del archivo haciendo uso de un for.