Tema 12:
Ficheros

Tema 12 Ficheros. Indice

1. Ficheros

- Surge ante las necesidad de almacenar información de forma permanente.
- Es un conjunto de información homogénea y relacionada, almacenada en un soporte permanente bajo un nombre.
- Un fichero no es una estructura de datos. Las estructuras de datos es la organización de datos en memoria volátil (RAM).

Tipos de ficheros

- Dependiendo de la dirección del flujo de datos en el programa:
  - De Entrada (o de Lectura): El programa lee los datos del archivo.
  - De Salida (o de Escritura):
  - De Entrada /Salida (Lectura/Escritura): Los datos pueden ser leídos y escritos en el fichero.
- El mismo fichero puede ser de entrada para un programa y de salida para otro.

Tipos de ficheros

- Dependiendo del contenido
  - **Ficheros de texto**: Contiene caracteres ASCCII. Solo están permitidos ciertos rangos de valores para cada byte. Existe un carácter especial que marca el fin del fichero
  - **Ficheros binarios**: Están permitidos todos los valores para cada byte. El fin de fichero se detecta de otro modo, dependiendo del soporte y del S.O (normalmente se hace guardando la longitud del fichero).
- Cuando queramos almacenar valores enteros, reales, imágenes, registros completos, etc, debemos usar ficheros binarios.

Tipos de ficheros

- Según el tipo de acceso en el programa
  - **Archivos de acceso secuencial**

    Para acceder a un dato del fichero es necesario acceder antes a todos los datos que se encuentran físicamente antes que él en el fichero.
  - **Archivos de acceso directo**

    Permiten acceder directamente a cualquier dato para realizar lecturas y/o escrituras.
- El mismo fichero, un programa puede tratarlo como de acceso secuencial y otro como de acceso directo.
- En sistemas de almacenamiento secuenciales (cintas magnéticas) no pueden crearse archivos de acceso directo.

2.Funciones básicas para operar con ficheros

- La E/S de ficheros en C se realizan a través de una serie de funciones de la stdio.h
- Las funciones son independientes del dispositivo (diskette, disco, impresora…)
- Cada dispositivo está asociado a un buffer.
- El buffer funciona haciendo de dispositivo lógico asociado a cualquier dispositivo físico (disco, diskette, impresora…)
- En C al comenzar cada programa se abren automáticamente los siguientes ficheros
  - stdin (Entrada estándar, por defecto el teclado)
  - stdout (salida estándar, por defecto la pantalla)
  - stderr (salida para errores, por defecto la pantalla)

Pasos para operar con ficheros

- **Crear un descriptor para el fichero** (sirve de indicador de posición del fichero)
  FILE * f;
- **Abrir el fichero**
  f= fopen (nombrefichero, modo_de_apertura);
- **Comprobar que se ha abierto correctamente**
  if (f == NULL)
  printf("Error en la apertura del fichero");
  else
  …….
- **Realizar operaciones de lectura/ escritura sobre fichero**
- **Cerrar el fichero** (cuando se ha abierto correctamente)
  fclose (f);

(Ver más funciones sobre ficheros en funcionesficheros.pdf)

Modos básicos de apertura

Para ficheros de texto:

- Lectura "r": Si se abre un fichero para lectura y no existe, devuelve NULL.
- Escritura "w": Si se abre un fichero para escritura y no existe se crea. Si ya existe se borrará el existente y se comienza a escribir uno nuevo
- Append (añadir) "a": Si se abre para añadir, si no existe se crea. Si existe se posicionará al final del fichero para leer o escribir.

Para ficheros binarios:

- Se añade "b". Es decir los modos básicos serían "rb", "wb", "ab".
- Utilizaremos también los modos de lectura/escritura ("rb+","wb+", "ab+")

3. Ficheros de texto

3.1. Funciones sobre caracteres

- Leer un carácter del fichero

  c = fgetc (f);

  Devuelve el carácter leído e incrementa el indicador de posición del archivo, apuntando al siguiente carácter a leer

- Escribir un carácter en un fichero

  fputc(c, f);

  Escribe el carácter leído en el fichero e incrementa el indicador de posición del fichero.

3.2. Funciones sobre cadenas de caracteres

- Leer una cadena de caracteres de un fichero
  fgets(char cad[], int n, FILE * f)

  Lee una cadena de caracteres de un fichero, hasta que encuentra un carácter de nueva línea o se hayan leído n-1 caracteres.
- Escribir una cadena de caracteres en un fichero
  fputs(char cad[], FILE * f)

  Escribe la cadena de caracteres en el fichero (sin escribir salto de línea)
- Ambas funciones incrementan el indicador de posición del fichero.

Recorrido secuencial de un fichero

```
FILE * f;
char c;
f= fopen("fichero.txt", "r");
if (f == NULL)
    printf("Error. No se ha podido abrir el fichero\n");
else
{
    c= fgetc(f);
    while ( feof (f) == 0)
    {
      // Tratatamiento al carácter leido
      c= fgetc(f);
    }
   fclose(f);
}
```

3.3. E/S con formato en ficheros de texto

- fprintf (FILE * f, cadena, argumentos…)

  Funciona exactamente igual que printf pero el primer parámetro indica el fichero donde se va a escribir
- fscanf(FILE * f ,cadena, argumentos…)

  Funciona exactamente igual que scanf pero el primer parámetro indica el fichero desde donde se va a leer

4. Argumentos de la función main

- Algunas veces necesitamos pasar argumentos que sirvan de entrada al programa principal (main)
- Si queremos que el main reciba argumentos debe declararse de esta forma

  void main (int argc, char * argv[])
  - **argc**: Es el número de parámetros en la línea de comando (incluido el nombre del programa)
  - **argv**: Es un array de cadena de caracteres que contiene los argumentos de forma que

    argv[0] : Nombre del programa

    argv[1]: Primer argumento

    argv[2]: Segundo argumento

    ….. etc

5. Tratamiento de ficheros binarios

- Los ficheros binarios pueden almacenar cualquier tipo de dato (no sólo caracteres)
- Normalmente utilizaremos ficheros binarios en los que la información está organizada en base a registros.
- Las funciones para leer/escribir en ficheros binarios permite leer/escribir bloques de datos de distintos tipos.

Operador sizeof

- Recibe como parámetro un tipo de dato y devuelve el número de bytes que ocupa este tipo de dato

```
#typedef struct
{
        char nombre[20];
        float sueldo;
}Emple;
…
printf("El tamaño del registro Empleado es %d", sizeof(Emple));
printf("El tamaño de un entero es %d", sizeof(int));
```

5.1. Funciones de E/S
Escribir en un fichero binario: fwrite

- fwrite (pinf, n_bytes, n_elem, FILE * f)
  - pinf: Puntero a la información que va a ser escrita
  - n_bytes: número de bytes que se van a escribir
  - n_elem: número de elementos que se van a escribir (normalmente 1)
  - f: Descriptor del fichero
- Escribe (a partir de la posición que indique el puntero pinf) el número de bytes que indique n_bytes* n_elem.

Ejemplo de uso de fwrite

```
FILE * f;
Emple remple; // registro de Empleado

f= fopen("ficherobinario.dat", "wb");
if (f == NULL)
   printf("Error. No se ha podido abrir el fichero\n");
else
{
   remple.sueldo=25000;
   strcpy (remple.nombre, "Andres Suarez");
   fwrite(&remple, sizeof(Emple), 1, f);
   fclose(f);
}
```

Leer de un fichero binario: fread

- fread (pinf, n_bytes, n_elem, FILE * f)
  - pinf: Puntero donde se almacenará la información que va a ser leída.
  - n_bytes: número de bytes que se van a leer
  - n_elem: número de elementos que se van a leer (normalmente 1)
  - f: Descriptor del fichero

Ejemplo de uso de fread

```
FILE * f;
Emple re; // registro de Empleado
f= fopen("ficherobinario.dat", "rb");
if (f == NULL)
   printf("Error. No se ha podido abrir el fichero\n");
else
{
   fread ( &re, sizeof(Emple),1, f)
   while (! feof(f))
   {
        printf("\n Empleado %s Sueldo % f", re.nombre, re.sueldo)
        fread ( &re, sizeof(Emple),1, f)
   }
   fclose(f);
}
```

5.2. Funciones para desplazamiento en ficheros

- Hasta ahora hemos visto tratamiento secuencial de ficheros, para llegar a un registro hay que pasar por todos los anteriores.
- En ficheros de acceso directo es posible situarse directamente en cualquier punto del fichero para leer y/o escribir.
- Para que el fichero sea de acceso directo debemos tener un campo que actúe como índice del registro dentro del fichero.

Ejemplo fichero de acceso directo

Fichero peliculas.dat: Por el código de película puedo saber su posición dentro del fichero

DR01	La vida es bella	1999
ES02	Todo sobre mi madre	2000
HI03	El señor de los anillos	2002
DR04	Algo para recordar	1998
CO05	Algo pasa con Mary	2001
HI06	Trescientos	2007

Ficheros de acceso directo

- Para que en un fichero pueda realizarse acceso directo a sus registros debe cumplirse:
  - Sus registros tienen que tener algún campo clave que identifique de forma única al registro
  - Debe existir una correspondencia directa entre los valores del campo clave y las direcciones lógicas en el soporte (es decir,entre el campo clave y la posición del registro en el fichero).

Función fseek

Sirve para colocar el indicador de posición del fichero en cualquier punto de éste, de forma que podamos leer y/o escribir

`fseek (f, desp, pos)`

- **`f`**: Indicador de posición del archivo
- **`pos`**: Posición relativa al fichero desde donde va a situarse
  - SEEK_SET: Principio del fichero
  - SEEK_CUR: Posición actual del indicador del fichero
  - SEEK_END: Final del fichero
- **`desp`**: Desplazamiento en bytes a partir de la posición indicada. Es de tipo long int

Ejemplo de uso de fseek

```
Para el fichero peliculas.dat (suponemos que ya se
  ha abierto correctamente el fichero en modo "rb")

fseek ( f,2*(sizeof(Pelicula), SEEK_SET)
fread(&rpeli, sizeof(Pelicula),1,f);
// Leería el registro 3º, "El señor de los anillos"

fseek ( f, -1*(sizeof(Pelicula), SEEK_END);
fread(&rpeli, sizeof(Pelicula),1,f);
//Leería el último registro, "Trescientos"
```

Función ftell

`ftell (f)`

Devuelve el número de bytes entre el principio del archivo y el indicador de posición.

Si f está al final del archivo esta función nos sirve para calcular el tamaño en bytes del fichero

(ver fbinario.cpp)

Ficheros de lectura /escritura

Modos de lectura / escritura

- "rb+": El fichero tiene que existir. El indicador de posición se pone al principio
- "wb+": Si el fichero existe lo borra y si no existe lo crea. El indicador de posición se pone al principio
- "ab+": Si no existe lo crea . El indicador de posición se pone al final

Ficheros de lectura /escritura

¡¡ IMPORTANTE !!

Entre una operación de lectura y otra de escritura siempre debe existir un fseek.

Ejemplo:

```
(Suponemos que el fichero "peliculas.dat" ya se ha
  abierto correctamente, p.ejem, modo "rb+")
//Cambiaría el año de la película "Algo para
  recordar" a 1999
fseek(f,3*sizeof(Pelicula),SEEK_SET);
fread(&rpeli,sizeof(Pelicula),1,f); //Leería la película
rpeli.anno=1999;
fseek(f,3*sizeof(Pelicula),SEEK_SET);
fwrite(&rpeli,sizeof(Empleado),1,f); // Escribiría el
  registro modificado
```

6.Ficheros binarios y tablas de registros

- Es posible volcar una tabla de registros completa en un fichero con una sola instrucción.
- Solo podremos hacer esto si no existen huecos en la tabla, ya que se vuelca la tabla completa

```
Socio tsocio[TAM];
.....
fwrite (& tsocio[0], sizeof(Registro), TAM, f);
//equivalente a:
fwrite (tsocio, sizeof(Registro), TAM, f);
```

Ficheros binarios y tablas de registros

- De forma idéntica también se puede leer un conjunto de registros de un fichero de una sola vez y guardarlo en una tabla de registros
- Tanto para leer como para escribir por bloques tengo que saber el número de registros que se quiere leer o escribir

```
Socio tsocio[TAM];
.....
fread (& tsocio[0], sizeof(Registro), TAM, f);
//equivalente a:
//fread (tsocio, sizeof(Registro), TAM, f);
```