LECCIÓN 4:Ficheros

María de la Paz Guerrero Lebrero
Curso 2014 / 2015
Grado en Matemáticas
maria.guerrero@uca.es

UCA
Universidad de Cádiz

Índice

- Introducción
- Tipos de ficheros
- Funciones para el uso de ficheros
  - Apertura
  - Lectura
  - Escritura
  - Cierre
  - Otras funciones

Introducción

- Hasta ahora no habíamos visto ninguna forma de guardar permanentemente los datos y resultados de nuestros programas.

- Los ficheros no son únicamente los archivos que guardamos en el disco duro, en C todos los dispositivos del ordenador se tratan como ficheros: la impresora, el teclado, la pantalla,...

Tipos de ficheros

- Secuenciales
- Acceso directo
- Declaración de ficheros

Ficheros secuenciales

- La forma más simple de estructura de archivo es el *archivo secuencial*.
- En este tipo de archivo, los registros se sitúan físicamente en el dispositivo en el orden en el que se van escribiendo, uno tras otro y sin dejar huecos entre sí.
- El acceso a los registros también debe hacerse en orden, de modo que para acceder al registro N es necesario pasar primero por el registro 1, luego por el 2, luego por el 3, y así hasta llegar al registro N.

Ficheros secuenciales

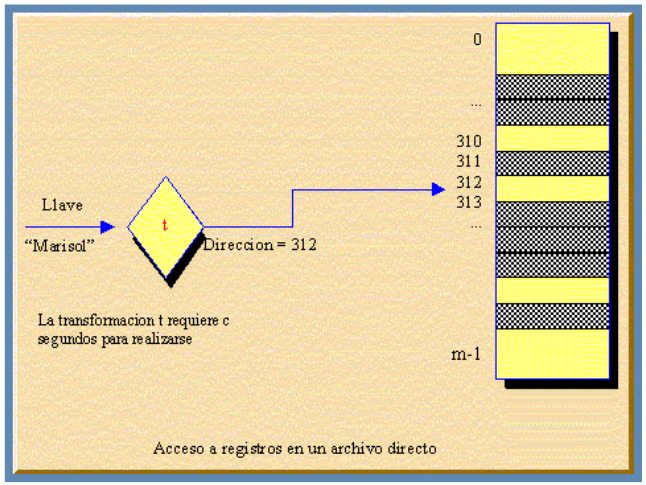
- Ventajas:
  - Es la más sencilla de manejar para el programador.
  - Si hay que acceder a un conjunto de registros consecutivos, o a todo el archivo, es el método más rápido.
  - No deja espacios entre registro y registro, por lo que se optimiza el uso del espacio en la memoria secundaria.

Ficheros secuenciales

- Inconvenientes:
  - Para consultar datos individuales, hay que recorrer todo el archivo desde el principio. Es decir, el acceso a registros individuales es, en general, lento.
  - Las operaciones de inserción y eliminación de registros solo pueden hacerse al final del archivo. Hacerlas con registros intermedios representa mover grandes bloques de información y, por lo tanto, consumir mucho tiempo.

Ficheros de acceso directo

- En este tipo de archivo, los registros se sitúan físicamente en el dispositivo sin ningún orden, dejando huecos entre sí.
- El acceso a los registros se realiza especificando directamente la posición dentro del fichero.



Ficheros de acceso directo

- Ventajas:
  - No hace falta recorrer todo el fichero para acceder a un registro.
  - Las operaciones de inserción y eliminación se pueden hacer en cualquier lugar del fichero.

Ficheros de acceso directo

- Inconvenientes:
  - Si hay que acceder a un conjunto de registros consecutivos, o a todo el archivo, hay que ir buscando cada uno de los registros. El proceso es más lento.
  - Deja espacios entre registro y registro, por lo que no se optimiza el uso del espacio en la memoria secundaria.

Declaración de ficheros

```
FILE *nombre_fichero;
```

- FILE es un puntero que apunta a una estructura que contiene información sobre el fichero.

- Todas las funciones de estrada/salida estándar usan este puntero para conseguir información sobre el fichero abierto

Funciones para el uso de ficheros

- Apertura
- Cierre
- Lectura
- Escritura
- Otras funciones

Apertura

fp = fopen (“nombre_fich”, “modo”);

- Devuelve un puntero al fichero abierto.
- El modo es la forma en la que se quiere abrir el fichero:

Modo	Significado
r	Sólo lectura
w	Escritura desde el comienzo del archivo
a	Escritura al final del archivo
r+	Lectura y escritura
w+	Escritura y lectura
rb	Sólo lectura (archivo binario)
wb	Escritura desde el comienzo del archivo (binario)
ab	Escritura el final del archivo (binario)

Cierre

```
int fclose (FILE *stream);
```

- Cierra un fichero que se ha abierto.
- Devuelve cero si el fichero ha sido cerrado correctamente, si ha habido algún error, devuelve la constante EOF.

Cierre

- Ejemplo de uso de la función *fclose:*

```
#include <stdio.h>
 int main() {
        FILE *fp;
        fp = fopen ( "fichero.in", "r" );
        fclose ( fp );
        system("pause");
}
```

Lectura

```
int fgetc(FILE *nombre_fich);
```

- Lee un carácter del fichero que se le pasa como parámetro.
- Si llega al final de fichero devuelve EOF.

Lectura

- Ejemplo de uso de la función *fgetc:*

```
#include <stdio.h>
#include <stdlib.h>
 int main()
{
                FILE *archivo;
               char caracter;
               archivo = fopen("prueba.txt","r");
                if (archivo == NULL)
                              exit(1);
               printf("\nEl contenido del archivo de prueba es \n\n");
               while (feof(archivo) == 0) {
                              caracter = fgetc(archivo);
                              printf("%c",caracter);
               }
 return 0;
}
```

Lectura

```
char* fgets(char *cad, int num, FILE *nombre_fich);
```

- Lee *num-1* caracteres del fichero y los almacena en la cadena *cad*.
- Si se encuentra el carácter de nueva línea o fin de fichero termina el proceso.
- Incluye el carácter '\0' en la cadena *cad*.

Lectura

- Ejemplo de uso de la función *fgets:*

```
int main()
{
        FILE *archivo;
        char  *caracteres;
        caracteres = (char*)malloc(100*sizeof(char));
        archivo = fopen("prueba.txt","r");
        if (archivo == NULL)
                exit(1);
        printf("\nEl contenido del archivo de prueba es \n\n");
        while (feof(archivo) == 0) {
                fgets(caracteres,100,archivo);
                puts(caracteres);
        }
 return 0;
}
```

Lectura

```
size_t fread (void *data, size_t size, size_t count, FILE *stream);
```

- En estas definiciones se usa el tipo **size_t**, el cuál está definido en *stddef.h* y sirve para definir tamaños de objetos.
- Recibe un puntero a donde almacenaremos los datos leídos, el tamaño de los datos a leer, la cantidad de esos datos a leer y el fichero.

Lectura

- Ejemplo de uso de la función *fread:*

```
int main ()
{
        FILE *fp;
        char *buffer;

        buffer = (char*)malloc(100*sizeof(char));
        fp = fopen ( "fichero.in", "r+" );

        fread ( buffer, sizeof ( char ), 100, fp );
        printf("%s", buffer); fclose ( fp );
        return 0;
}
```

Lectura

- **EJERCICIO 1:** Implementa un programa que pida al usuario el nombre de un fichero y un carácter. El programa debe leer el contenido del fichero y mostrarlo por pantalla cambiando el carácter que el usuario introdujo por el carácter '$'.

Escritura

```
int fputc(int carácter, FILE *nom_fich);
```

- Escribe un carácter cada vez en el fichero que se le pasa como parámetro.
- Devuelve el carácter escrito, si la operación fue completada con éxito, en caso contrario será EOF.

Escritura

- Ejemplo de uso de la función *fputc:*

```
int main ()
 {
            FILE *fp;
            char caracter;
            fp = fopen ( "fichero.txt", w );
            printf("\nIntroduce un texto en el fichero: ");
           while((caracter = getchar()) != '\n') {
                        printf("%c", fputc(caracter, fp));
           }
            fclose ( fp );
           return 0;
}
```

Escritura

```
int fputs(const char *buffer, FILE *nom_fich);
```

- 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 el fichero donde se realizará la escritura.

Escritura

- Ejemplo de uso de la función *fputs:*

```
#include <stdio.h>
 int main ()
{
          FILE *fp;
         char cadena[] = "Mostrando el uso de fputs en un
                                fichero.\n";
          fp = fopen ( "fichero.txt", "r+" );
         fputs( cadena, fp );
         fclose ( fp );
         return 0;
}
```

Escritura

```
size_t fwrite(void *puntero, size_t tamaño, size_t cantidad, FILE *archivo);
```

- Esta función está pensada para trabajar con registros de longitud constante y forma pareja con **fread**.
- Escribe en un fichero uno o varios registros de la misma longitud almacenados a partir de una dirección de memoria determinada.
- Devuelve 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
  - El fichero del que se hará la lectura

Escritura

- Ejemplo de uso de la función *fwrite:*

```
#include <stdio.h>

void menu();
void CrearFichero(FILE *Fichero);
 void InsertarDatos(FILE *Fichero);
 void VerDatos(FILE *Fichero);

struct sRegistro {
        char Nombre[25];
        int Edad;
        float Sueldo;
} *registro;
```

Escritura

```
int main() {
        int opcion;
        int exit = 0;
        FILE *fichero;
        while (!exit) {
                menu();
                printf("\nOpcion: ");
                scanf("%d", &opcion);
                switch(opcion) {
                        case 1: CrearFichero(fichero); break;
                        case 2: InsertarDatos(fichero); break;
                        case 3: VerDatos(fichero); break;
                        case 4: exit = 1; break;
                        default: printf("\nopcion no valida");
                }
        }
        return 0;
}
```

Escritura

```
void menu()
 {
        printf("\nMenu:");
        printf("\n\t1. Crear fichero");
        printf("\n\t2. Insertar datos");
        printf("\n\t3. Ver datos");
        printf("\n\t4. Salir");
}
```

Escritura

```
void CrearFichero(FILE *Fichero)
{
        Fichero = fopen("fichero.txt", "r");
         if(!Fichero) {
                Fichero = fopen("fichero.txt", "w");
                printf("\nArchivo creado!");
        }
         else {
                printf("\nEl fichero ya existe!");
        }
        fclose (Fichero);
}
```

Escritura

```
void InsertarDatos(FILE *Fichero) {
        Fichero = fopen("fichero", "r+");
         if(Fichero == NULL) {
                printf("\nFichero no existe! \nPor favor creelo");
                exit(1);
        }
        printf("\nDigita el nombre: ");
        scanf("%s", &registro.Nombre);
        printf("\nDigita la edad: ");
        scanf("%d", &registro.Edad);
        printf("\nDigita el sueldo: ");
        scanf("%f", &registro.Sueldo);
        fwrite(registro, sizeof(struct sRegistro), 1, Fichero);
        fclose(Fichero);
}
```

Escritura

```
void VerDatos(FILE *Fichero) {
        int numero = 1;
        Fichero = fopen("fichero", "r");
        if(Fichero == NULL) {
                printf("\nFichero no existe! \nPor favor creelo");
                return;
        }
        fread(registro, sizeof(struct sRegistro), 1, Fichero);
        printf("\nNumero \tNombre \tEdad \tSueldo");
        while(!feof(Fichero)) {
                printf("\n%d \t%s \t%d \t%.2f", numero, registro.Nombre,
                        registro.Edad, registro.Sueldo);
                fread(registro, sizeof(struct sRegistro), 1, Fichero);
                numero++;
        }
         fclose(Fichero);
}
```

Escritura

- **EJERCICIO 2:** Implementa un programa que pida al usuario el nombre de dos ficheros, un carácter y su nombre. El programa debe leer el contenido del primer fichero y escribirlo en el segundo cambiando el carácter que el usuario introdujo por el carácter '$'. Al final debe escribir el nombre del usuario en el segundo fichero.

Otras funciones

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

- Determina si el cursor del archivo encontró el final del mismo(**e**nd **o**f **f**ile).
- Devuelve cero (Falso) si no encuentra el final del fichero, de lo contrario devolverá un valor distinto de cero (Verdadero).

Otras funciones

- Ejemplo de uso de la función *feof:*

```
int contar_caracteres ()
{
  FILE * f;
  int  numero = 0;
  char caracter;

  f = fopen ("texto.txt", "r");

  if (f == NULL)
    return -1;

  while (feof (f) == 0)
  {
    fscanf (f, "%c", &caracter);
    if (caracter == clave)
      numero++;
  }

  fclose (f);

  return numero;
}
```

Otras funciones

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

- Sitúa el cursor de lectura/escritura al principio del fichero.
- No devuelve nada

Otras funciones

- Ejemplo de uso de la función *rewind:*

```
#include <stdio.h>

int main ()
{
   int n;
   FILE * pFile;
  char *buffer;
  pFile = fopen ("myfile.txt","w+");
  for ( n='A' ; n<='Z' ; n++)
      fputc ( n, pFile);
  rewind (pFile);
  buffer = (char*)malloc(27);
  fread (buffer,1,26,pFile);
  fclose (pFile);
  buffer[26]='\0';
  puts (buffer);
  return 0;
}
```

Otras funciones

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

- Desplaza la posición actual de lectura/escritura del fichero a otro punto.
- El desplazamiento puede ser positivo (avanzar), cero o negativo (retroceder).
- La posición de origen se puede indicar con la ayuda de tres constantes: SEEK_SET (0, comienzo), SEEK_CUR (1, actual), SEEK_END (2, final)

Otras funciones

- Ejemplo de uso de la función *fseek:*

```
#include <stdio.h>

int main ()
{
  FILE * pFile;
  pFile = fopen ( “ejemplo.txt" , "w" );
  fputs ( “Esto es un ejemplo." , pFile );
  fseek ( pFile , 9 , SEEK_SET );
 fputs ( " hola" , pFile );
 fclose ( pFile );
 return 0;
}
```

Otras funciones

```
long int ftell ( FILE * stream );
```

- Devuelve el valor actual del indicador de posición del fichero.
- Para ficheros binarios devuelve el número e bytes desde el comienzo del fichero.
- Para ficheros de texto devuelve la posición del cursor que puede ser utilizada por la función **fseek**.

Otras funciones

- Ejemplo de uso de la función *ftell:*

```
#include <stdio.h>
int main () {
   FILE * pFile;
   long size;
   pFile = fopen (“fichero.txt","rb");
   if (pFile==NULL)
      printf ("Error , no se puede abrir el fichero\n");
   else {
      fseek (pFile, 0, SEEK_END);
      size=ftell (pFile);
      fclose (pFile);
      printf (“Tamaño de fichero.txt: %ld bytes.\n",size);
   }
   return 0; }
```

Otras funciones

```
int fprintf ( FILE * stream, const char * format, ... );
```

- La función *fprintf* funciona igual que *printf* en cuanto a parámetros, pero la salida se dirige a un fichero cuyo manejador se especifica como primer parámetro en la función en lugar de a la pantalla.

Otras funciones

- Ejemplo de uso de la función *fprintf:*

```
int main () {
  FILE * pFile;
  int n;
  char *name;
  pFile = fopen (“mifichero.txt","w"); //Se comprueba su apertura
  for (n=0 ; n<3 ; n++) {
     printf (“Escriba un nombre: ");
    name = InicializarCadena();
    fprintf (pFile, “%s\n",name);
  }
 fclose (pFile);
 return 0;
}
```

Otras funciones

```
int fscanf ( FILE * stream, const char * format, ... );
```

- La función *fscanf* funciona igual que *scanf* en cuanto a parámetros, pero la entrada se toma de un fichero cuyo manejador se especifica como primer parámetro en lugar del teclado.

Otras funciones

- Ejemplo de uso de la función *fscanf:*

```
int main () {
  char *str;
  float f;
  FILE * pFile;
  pFile = fopen ("myfile.txt","w+");  //Se comprueba su apertura
  fprintf (pFile, "%f %s", 3.1416, "PI");
  rewind (pFile);
  fscanf (pFile, "%f", &f);
  str = (char*)malloc(2*sizeof(char));
  fscanf (pFile, "%s", str);
  fclose (pFile);
  printf ("Contenido: %f and %s \n", f, str);
  return 0;
}
```

Otras funciones

- **EJERCICIO 3:** Implementa un programa que le pida al usuario el nombre de un fichero y 12 números (del 1 al 12) **en cualquier orden y sin repeticiones**. El programa debe almacenar en el fichero los 12 números separados por un espacio en blanco. A continuación, se le pide al usuario un número *n* (del 1 al 12), se lee del fichero el número *m* que está en la posición *n* y se devuelve como resultado el valor que hay en la posición *m*.

- Por ejemplo, si el contenido del fichero es: **8 3 12 7 9 11 5 2 10 1 6 4**, y el valor *n* = *8* implica que *m* = *2*, y por tanto, devolvería 3.