INFORMÁTICA APLICADA

TEMA: Ficheros
(Notas de clase)

Aviso: este documento es sólo un resumen de los conceptos básicos del tema. Es muy recomendable ampliarlos con la lectura de algún libro de la bibliografía de la asignatura.

1.- Introducción

El almacenamiento de datos en variables es temporal; al terminar un programa todos estos datos se pierden. Para la conservación permanente de grandes cantidades de datos se utilizan los *ficheros*. Los computadores almacenan los ficheros en dispositivos de almacenamiento secundario, especialmente en dispositivos de almacenamiento en disco. Los ficheros almacenados en estos dispositivos deben tener un nombre válido para el sistema operativo en el que se esté trabajando. Estos ficheros están controlados directamente por el sistema operativo, y nuestro programa lo único que hace es solicitar los servicios del mismo para el manejo de nuestros datos.

En este tema vamos a estudiar cómo se crean, actualizan y procesan ficheros de datos en general, y utilizando el lenguaje C en particular.

2.- Tipos de ficheros

Dependiendo del tipo de información guardada en el fichero, y de su interpretación, existen dos tipos: *de texto* y *binarios*.

- Los ficheros ***de texto*** almacenan caracteres.
- Los ficheros ***binarios*** almacenan bytes y trabajan en bloques de un determinado tamaño.

La principal diferencia entre los dos tipos es que los caracteres que se envían a un fichero de texto pueden sufrir una traducción. Por ejemplo, el carácter de fin de línea (‘\n’) se transforma en una combinación de caracteres, el carácter <CR> (llamado “retorno de carro”) y el carácter <LF> (llamado “avance de línea”), mientras que en los binarios no se produce esta traducción. Un fichero binario es una copia exacta de los datos que se encuentran en la memoria del ordenador, pero en un dispositivo de almacenamiento secundario. Dependiendo del tipo de fichero con el que se trabaje, se utilizarán distintas funciones en su procesamiento.

Según el tipo de acceso, podemos distinguir dos categorías de ficheros: *de acceso secuencial* y *de acceso directo*.

- Cuando los datos de un fichero se leen y escriben uno a continuación de otro, se dice que el fichero es ***de acceso secuencial***.
- Si se accede a cualquier dato del fichero directamente, independientemente de su posición en el mismo, se dice que el fichero es ***de acceso directo***.

En el lenguaje C un fichero se ve simplemente como un flujo secuencial de bytes. Todos los ficheros terminan con un marcador de fin de fichero (eof, *end of file*). Cuando un fichero *se abre*, se asocia un flujo con ese fichero. En la apertura de un fichero se obtiene un puntero a una estructura llamada FILE (definida en <stdio.h>), que contiene información necesaria para procesar el fichero. La librería estándar de C proporciona muchas funciones para leer y escribir datos en los ficheros que se tengan abiertos.

Para trabajar con un fichero en un programa se suele seguir el siguiente esquema:

1) Declaración de la variable fichero
2) Apertura del fichero
3) Lectura / escritura en el fichero
4) Cierre del fichero

En las siguientes secciones vamos a describir cada uno de estos pasos.

3.- Declaración de la variable fichero

Siempre que se trabaje con ficheros en C se debe declarar una variable de tipo fichero, que en realidad será un puntero a una estructura FILE con información sobre el fichero. A esta variable se le asignará posteriormente un fichero del disco, y se utilizará como identificador del mismo para realizar las operaciones. La sintaxis para declarar la variable fichero en C es:

```
FILE *variableFichero;
```

Como ya se ha dicho, FILE es una estructura que está declarada en el fichero de cabecera stdio.h, y por tanto debe incluirse en el programa.

4.- Apertura de un fichero

La apertura de un fichero consiste en asignar a una variable de tipo FILE *, un fichero real, almacenado en un dispositivo externo. Para abrir un fichero se utiliza la función estándar fopen, cuya sintaxis puede verse en la transparencia 3. Por ejemplo, si se desea abrir el fichero C:\MiFichero.txt para leer, se escribirían sentencias como las siguientes:

```
FILE *f;
f = fopen("C:\\MiFichero.txt", "r");
```

Esta función devuelve un puntero para poder manejar el fichero en disco. Si no se puede abrir el fichero por cualquier problema, la función devuelve el puntero nulo (NULL). Es conveniente, por tanto, comprobar siempre si el fichero se ha abierto correctamente, con un sentencia como la siguiente:

```
FILE *f;
f = fopen(“C:\\MiFichero.txt”, “r”);
if (f == NULL)
{
      printf(“Error al abrir el fichero\n”);
      exit(1); // termina el programa
}
```

5.- Lectura y escritura en un fichero

Después de abrir el fichero se pueden realizar operaciones de lectura y escritura sobre el mismo. Existen varias funciones en C para realizar estas operaciones. En todas ellas se debe utilizar el identificador del fichero sobre el que se quiere leer o escribir. En las transparencias 4, 5, 6 y 7 se describen algunas de estas funciones junto con algunos ejemplos sencillos.

Para comprobar si se ha llegado al final del fichero se puede utilizar la función feof cuyo prototipo aparece en la transparencia 4. Para leer una cadena de caracteres puede utilizarse la función fgets, cuyo prototipo aparece en la transparencia 5. Esta función, al igual que muchas otras de tratamiento de ficheros, devuelve el puntero NULL si se ha producido algún error al leer del fichero. Por tanto, es muy conveniente comprobar el valor de retorno de estas funciones. La función fgets lee una cadena de caracteres del fichero especificado como tercer parámetro, almacenándola en la cadena pasada como primer parámetro. Los caracteres los va leyendo uno a uno hasta que se alcanza un carácter de nueva línea (‘\n’), o bien hasta que se hayan leído n-1 caracteres, siendo n el segundo parámetro. El carácter ‘\n’ también se incluye en la cadena, y al final se añade el carácter ‘\0’ que indica el fin de la cadena. Puede verse una descripción detallada de esta función en la ayuda del compilador.

Ejemplo: El siguiente bucle lee todas las líneas del fichero cuyo identificador es f, imprimiéndolas en pantalla:

```
while( !feof(f) )
    if ( fgets(linea, 100, f) != NULL )
      printf(“%s”, linea);
```

En las transparencias aparecen los prototipos de otras funciones de entrada/salida para ficheros:

- ***fseek***: se utiliza para acceder a una posición concreta del fichero.
- ***fprintf***: imprime con formato en un fichero, de forma análoga a como lo hace printf con la salida estándar.
- ***fscanf***: lee con formato de un fichero, de forma análoga a como lo hace scanf de la entrada estándar.
- ***fputc***: escribe un carácter en un fichero.
- ***fgetc***: lee un carácter de un fichero.
- ***fputs***: escribe una cadena en un fichero.
- ***fread***: lee bloques de bytes de un fichero.

- ***fwrite***: escribe bloques de bytes en un fichero.

En la ayuda del compilador de C++ de las aulas de prácticas puede encontrarse información completa sobre el comportamiento, parámetros y valores de retorno de cada una de estas funciones. Se recomienda consultar esta ayuda.

6.- Cierre de un fichero

Después de procesar el fichero, la operación de *cierre* del mismo garantiza que todos los datos se almacenarán correctamente. Esta operación actualiza los datos del fichero físicamente en el dispositivo y desliga el fichero del identificador. Para cerrar un fichero en C se utiliza la función fclose:

```
fclose(variableFichero);
```

Es muy conveniente cerrar todos los ficheros cuando se han terminado de procesar. Cuando un programa se termina, se cierran automáticamente todos los ficheros que mantenga abiertos, pero es una buena práctica de programación cerrarlos explícitamente en el programa con la función fclose.

7.- Ficheros estándar

En la mayoría de lenguajes de programación los dispositivos de entrada/salida son tratados como ficheros de texto. Estos dispositivos se consideran ficheros predefinidos que no necesitan ser abiertos por el programador y que pueden procesarse de igual forma que cualquier otro fichero. En C existen variables fichero predefinidas para algunos dispositivos de entrada/salida del computador:

- ***stdin*** corresponde a la entrada estándar (por defecto, el teclado).
- ***stdout*** corresponde a la salida estándar (por defecto, la pantalla).
- ***stderr*** corresponde a la salida de error estándar (por defecto, la pantalla).
- ***stdprn*** corresponde a la impresora estándar que tenga por defecto el sistema.

Por ejemplo, las sentencias siguientes son equivalentes:

```
fscanf(stdin,"%d",&i);      ⇔      scanf("%d",&i);
fputs(cadena, stdout);      ⇔      puts(cadena);
```

Al empezar la ejecución de un programa, automáticamente se abren tres archivos: la entrada estándar, la salida estándar y el error estándar. Estos tres dispositivos se asocian a los identificadores stdin, stdout y stderr.