Tema 5. Almacenamiento Persistente de Datos

Vicente Benjumea García

Programación-I
Departamento de Lenguajes y Ciencias de la Computación.
E.T.S.I. Informática. Univ. de Málaga.

Tema 5. Almacenamiento Persistente de Datos

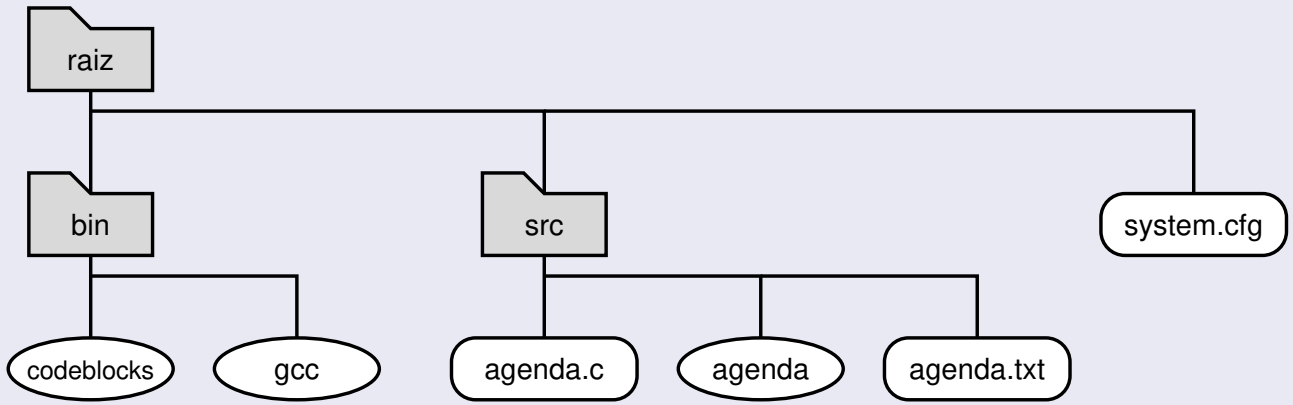
- Introducción. Conceptos básicos.
- Entrada y Salida de Datos Asociadas a Ficheros.
- Copiar un Fichero de Texto.
- Entrada de Datos desde Ficheros de Texto.
- Salida de Datos a Ficheros de Texto.

Introducción. Conceptos básicos (I)

- Almacenamiento de Datos en el Ordenador
  - Memoria Principal: (*acceso directo de la CPU*).
    - Tiempo de acceso muy rápido.
    - Almacenamiento **no persistente**: **volátil**.
    - Capacidad de almacenamiento limitada.
  - Memoria Secundaria: (*discos duros, discos ópticos, memorias USB, etc.*)
    - Tiempo de acceso lento.
    - Almacenamiento **persistente**.
    - Gran capacidad de almacenamiento.

Introducción. Conceptos básicos (II)

- Organización de la Memoria Secundaria (*gran capacidad de almacenamiento*)
  - Sistema de Ficheros
    - Jerarquía de Directorios (Carpetas) y Ficheros
    - Directorios: organizan jerarquicamente el sistema de ficheros
      Directorios, subdirectorios y ficheros
    - Ficheros: almacenamiento persistente de información
      Datos: información, configuraciones, código fuente
      Software: bibliotecas y programas ejecutables



Introducción. Conceptos básicos (III)

- La **entrada de datos** (lectura/cargar) se refiere a los datos que recibe el programa.
- La **salida de datos** (escritura/guardar) se refiere a los datos que el programa envía.
  - Ya hemos visto la entrada de teclado y la salida a pantalla.
  - Ahora vamos a tratar la entrada/salida con **ficheros**, almacenados en memoria secundaria, para el **almacenamiento de datos de forma persistente**.

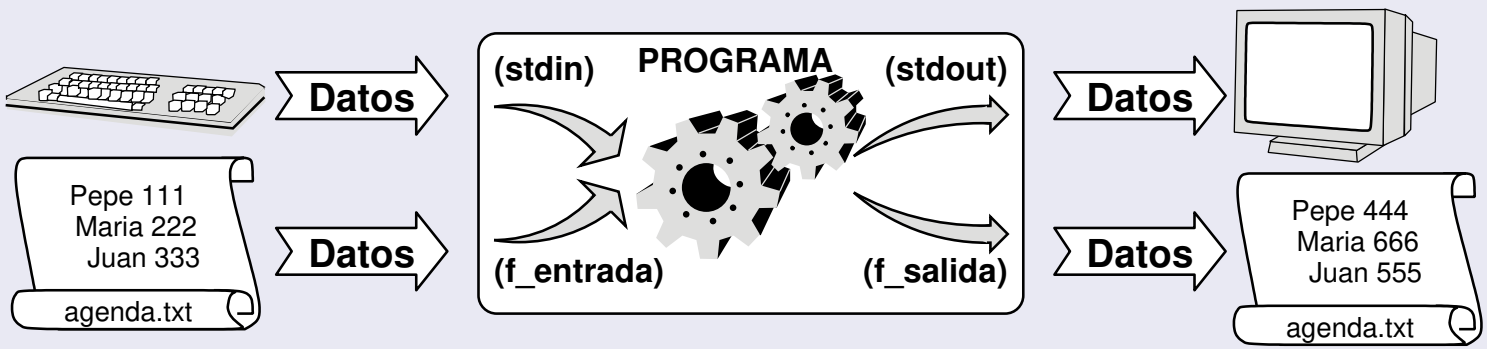
- Almacenamiento de la Información. Tipos de Ficheros:
  - Ficheros de Texto
    - Codificación textual: secuencia de caracteres (ASCII/UTF-8/etc) *(Conversión)*
    - Procesamiento orientado a ordenador (también humano)
    - Representan información muy diversa (editor de textos)
  - Ficheros Binarios
    - Codificación binaria: secuencia de bytes (rep. interna del ordenador)
    - Procesamiento orientado a ordenador (problemas de compatibilidad)
    - Representan información binaria (programas, imágenes, música, etc.)

Entrada y Salida de Datos Asociadas a Ficheros (I)

- Los ficheros (archivos) permiten **almacenar la información** de forma **permanente** en el sistema de almacenamiento.
- Un fichero contiene cierta información **codificada**, que se almacena en un memoria como una **secuencia de bytes**.
- Cada fichero recibe un **nombre** (posiblemente con una extensión) y se ubica dentro de un **directorio** (carpeta) que forma parte de una cierta **jerarquía** (ruta, camino o vía de acceso).
- El **nombre** y la **ruta**, o secuencia de directorios, que hay que atravesar para llegar a la ubicación de un fichero, **identifican** a dicho fichero de forma única dentro del sistema de ficheros.
  - En **Windows** se utiliza la *barra invertida* \ como carácter separador en cadenas de caracteres, En el programa se debe duplicar \\, ya que representa el símbolo de escape de los caracteres.
  - En **Unix**, **Linux** y **MacOSX**, se utiliza el símbolo / para separar los componentes de la ruta del fichero.
  - Para identificar los **ficheros en el directorio de trabajo**, solo se debe especificar el nombre del fichero, no es necesario utilizar el caracter separador.

Entrada y Salida de Datos Asociadas a Ficheros (II)

- Entrada y Salida de Datos a Través de Flujos (Streams)
  - Flujos estándares de entrada (**stdin**) y salida (**stdout**).
  - **Flujo de entrada**: *fuente* que proporciona una secuencia de caracteres
  - **Flujo de salida**: *sumidero* que recibe una secuencia de caracteres
  - Un **manejador de fichero**: *variable* que vincula un flujo de datos con un determinado fichero
  - Toda la transferencia de información se realiza a través de los manejadores de ficheros (a través de *buffers*).



Entrada y Salida de Datos Asociadas a Ficheros (III)

Operaciones con ficheros

- **Apertura**: establece una *conexión* entre un *manejador del fichero* dentro del programa y un determinado fichero dentro del sistema de almacenamiento. En caso de apertura para lectura (entrada) (**"r"**), el fichero debe existir. En caso de apertura para escritura (salida) (**"w"**), se borrará o creará un nuevo fichero.
- **Escritura**: para poder almacenar información en un fichero, una vez abierto en modo de escritura, hay que transferir la información **organizada** de alguna forma mediante operaciones de escritura.
- **Lectura**: para poder utilizar la información contenida en un fichero, debe estar abierto en modo de lectura, y hay que utilizar las operaciones de lectura adecuadas a la **organización** de la información contenida en dicho fichero.
- **Cierre**: cuando se ha terminado de transferir la información a o desde el fichero, se debe **cerrar** la conexión previamente establecida entre la variable manejador del fichero y el fichero en el sistema de almacenamiento. Esta operación se ocupa, además, de mantener la **integridad** del fichero, escribiendo previamente la información que se encuentre en algún buffer intermedio en espera de pasar al fichero. En caso de **no cerrar adecuadamente** el manejador de fichero, entonces es posible que se **pierdan** *recursos del sistema*, que el fichero **no** guarde correctamente todos los datos enviados, y que no se pueda garantizar la **integridad** del fichero.

Entrada y Salida de Datos Asociadas a Ficheros (IV)

Operaciones con ficheros (I)

- `FILE* fopen(const char* nombre, const char* modo)`
  abre el fichero cuyo nombre se especifica, en el modo especificado (**"r"**, **"w"**). Devuelve el manejador del fichero (de tipo `FILE*`), en caso de error devuelve `NULL`.
- `int fclose(FILE* mf)`
  cierra adecuadamene el manejador del fichero, abierto previamente con `fopen()`. Si la operación es correcta, devuelve el valor cero (0), y negativo en caso de error.
- `int fprintf(FILE* mf, const char* formato, ...)`
  escribe (guarda) en el fichero, la salida de texto, según el formato y datos (argumentos) especificados. Si la operación es correcta, devuelve el número de caracteres escritos, y negativo en caso de error.
- `int fscanf(FILE* mf, const char* formato, ...)`
  extrae (lee/carga) del fichero la entrada de texto, según el formato y datos (argumentos) especificados. Si la operación es correcta, devuelve el número de datos extraídos, y cero o negativo en caso de error.

Entrada y Salida de Datos Asociadas a Ficheros (V)

Operaciones con ficheros (II)

- `int feof(FILE* mf)`
  Si se ha llegado al final del fichero, devuelve el valor `true` (distinto de cero). Si **no** se ha llegado al final del fichero, devuelve el valor `false` (igual a cero).
- `int ferror(FILE* mf)`
  Si el estado del manejador del fichero es erróneo, devuelve el valor `true` (distinto de cero). Si el estado del manejador del fichero es correcto, devuelve el valor `false` (igual a cero).

Operaciones con ficheros (III)

- `char* fgets(char* linea, int tamaño, FILE* mf)`
  extrae (lee/carga) del fichero una línea de texto (hasta salto-de-línea `\n`), sin sobrepasar el *tamaño* especificado, y la almacena en la zona de memoria apuntada por *linea* (debe haber al menos el espacio indicado por tamaño). El carácter `\n` también se almacena en la zona de memoria. Si la operación es correcta, devuelve el puntero a la zona de memoria, y `NULL` en caso de error.

Entrada y Salida de Datos Asociadas a Ficheros (VI)

- Procesamiento General de Ficheros en un Programa C
  1. Incluir biblioteca `<stdio.h>`
  2. Declarar variable (manejador de fichero) de tipo `FILE*`
  3. Abrir el manejador de fichero: vincula el flujo con fichero (`fopen()`)
  4. Comprobar apertura correcta (`NULL`)
  5. Transferencia de información (`fprintf()`, `fscanf()`, `fgets()`)
  6. Comprobar transferencia de información correcta (`NULL`, `feof()`, `ferror()`)
  7. Cerrar el manejador de fichero: desvincula el flujo con el fichero (`fclose()`)

Copiar un Fichero de Texto (por caracteres) (I)

```c
#include <stdio.h>
#include <stdbool.h>
bool copiar_fich(const char* fich_org, const char* fich_dst)
{
    bool ok = false;
    FILE* f_ent = fopen(fich_org, "r");          // Apertura del Flujo de Entrada
    if ( f_ent != NULL ) {                       // ¿ Apertura Correcta ?
        FILE* f_sal = fopen(fich_dst, "w");      // Apertura del Flujo de Salida
        if ( f_sal != NULL ) {                   // ¿ Apertura Correcta ?
            char caracter ;
            while (( fscanf(f_ent, "%c", &caracter) == 1 ) && ! ferror(f_sal) ) {
                fprintf(f_sal, "%c", caracter) ;
            }
            ok = feof(f_ent) && ! ferror(f_sal) ;        // ¿ Correcto ?
            fclose(f_sal) ;                              // Cierre del Flujo de Entrada
        }
        fclose(f_ent) ;                                  // Cierre del Flujo de Entrada
    }
    return ok;
}
int main()
{
    bool ok = copiar_fich("datos1.txt", "datos2.txt") ;
    if ( ! ok ) {
        printf("Error en copia de fichero\n");
    }
}
```

Copiar un Fichero de Texto (por líneas) (II)

```
#include <stdio.h>
#include <stdbool.h>
enum { MAXCARS = 256 };
bool copiar_fich(const char* fich_org, const char* fich_dst)
{
    bool ok = false;
    FILE* f_ent = fopen(fich_org, "r");          // Apertura del Flujo de Entrada
    if ( f_ent != NULL ) {                       // ¿ Apertura Correcta ?
        FILE* f_sal = fopen(fich_dst, "w");      // Apertura del Flujo de Salida
        if ( f_sal != NULL ) {                   // ¿ Apertura Correcta ?
            char linea[MAXCARS] ;
            while (( fgets(linea, MAXCARS, f_ent) != NULL ) && ! ferror(f_sal) ) {
                fprintf(f_sal, "%s", linea) ;
            }
            ok = feof(f_ent) && ! ferror(f_sal) ;        // ¿ Correcto ?
            fclose(f_sal) ;                              // Cierre del Flujo de Entrada
        }
        fclose(f_ent) ;                                  // Cierre del Flujo de Entrada
    }
    return ok;
}
int main() {
    bool ok = copiar_fich("datos1.txt", "datos2.txt") ;
    if ( ! ok ) {
        printf("Error en copia de fichero\n");
    }
}
```

Entrada de Datos desde Ficheros de Texto (I)

```c
#include <stdio.h>
#include <stdbool.h>

bool leer_fich(const char* nombre_fichero, struct ListaNum* dat)
{
    bool ok = false;
    FILE* f_ent = fopen(nombre_fichero, "r");              // Apertura del Flujo de Entrada
    if ( f_ent != NULL ) {                                 // ¿ Apertura Correcta ?
        int numero ;
        while ( fscanf(f_ent, " %d", &numero) == 1 ) {     // ¿ Lectura Correcta ?
            procesar_numeros(dat, numero) ;
        }
        ok = feof(f_ent) ;                                 // ¿ Fin de Fichero ? (End-of-File)
        fclose(f_ent) ;                                    // Cierre del Flujo de Entrada
    }
    return ok;
}
int main()
{
    struct ListaNum dat = {};
    bool ok = leer_fich("datos.txt", &dat) ;
    if ( ! ok ) {
        printf("Error procesamiento de fichero\n");
    }
}
```

Entrada de Datos desde Ficheros de Texto (II)

```c
#include <stdio.h>
#include <stdbool.h>
enum { MAXCARS = 63+1 };
bool leer_fich(const char* nombre_fichero, struct Datos* dat)
{
    bool ok = false;
    FILE* f_ent = fopen(nombre_fichero, "r");              // Apertura del Flujo de Entrada
    if ( f_ent != NULL ) {                                 // ¿ Apertura Correcta ?
        char nombre[MAXCARS];
        int edad;
        double nota;
        while (fscanf(f_ent, " %63[^;\n] ; %d ; %lg", nombre, &edad, &nota) == 3) { // ¿ Lect
            procesar_datos(dat, nombre, edad, nota) ;
        }
        ok = feof(f_ent) ;                                 // ¿ Fin de Fichero ? (End-of-File)
        fclose(f_ent) ;                                    // Cierre del Flujo de Entrada
    }
    return ok;
}
int main()
{
    struct Datos dat = {};
    bool ok = leer_fich("datos.txt", &dat) ;
    if ( ! ok ) {
        printf("Error procesamiento de fichero\n");
    }
}
```

Salida de Datos a Ficheros de Texto (I)

```c
#include <stdio.h>
#include <stdbool.h>

bool escribir_fich(const char* nombre_fichero, const struct ListaNum* dat)
{
    bool ok = false ;
    FILE* f_sal = fopen(nombre_fichero, "w");  // Apertura del Flujo de Salida
    if ( f_sal != NULL ) {                     // ¿ Apertura Correcta ?
        for (int i = 0; (i < dat->nelms) && ! ferror(f_sal); ++i) { // ¿ Hay Datos ? ¿ Flujo
            fprintf(f_sal, "%d\n", dat->elm[i]);    // Salida de datos (SEPARADORES ADECUADOS
        }
        ok = ! ferror(f_sal) ;                 // ¿ Salida Correcta ?
        fclose(f_sal) ;                        // Cierre del Flujo de Salida
    }
    return ok;
}
int main()
{
    struct ListaNum dat = { ... };
    bool ok = escribir_fich("datos.txt", &dat) ;
    if ( ! ok ) {
        printf("Error escritura de fichero\n");
    }
}
```

Salida de Datos a Ficheros de Texto (II)

```
#include <stdio.h>
#include <stdbool.h>
struct Persona {
    char nombre[MAXCARS];
    int edad;
    double nota;
};
struct Datos {
    int nelms;
    struct Persona elm[MAXELMS];
};
bool escribir_fich(const char* nombre_fichero, const struct Datos* dat)
{
    bool ok = false ;
    FILE* f_sal = fopen(nombre_fichero, "w");  // Apertura del Flujo de Salida
    if ( f_sal != NULL ) {                     // ¿ Apertura Correcta ?
        for (int i = 0; (i < dat->nelms) && ! ferror(f_sal); ++i) { // ¿ Hay Datos ? ¿ Flujo
            fprintf(f_sal, "%s; %d; %lg\n",
                           dat->elm[i].nombre, dat->elm[i].edad, dat->elm[i].nota);
                           // Salida de datos (SEPARADORES ADECUADOS)
        }
        ok = ! ferror(f_sal) ;                 // ¿ Salida Correcta ?
        fclose(f_sal) ;                        // Cierre del Flujo de Salida
    }
    return ok;
}
```