Entrada / Salida en C

Antonio Espín Herranz

Entrada / Salida

- Teclado y consola.
- Tratamiento de ficheros.
- Lectura y escritura de ficheros
  - De tipos básicos.
  - De tipos definidos por el usuario.
- Cadenas de caracteres.
- Funciones estándar sobre cadenas.
- Cadenas como punteros.

Introducción

- Un flujo (stream) es una abstracción que se refiere a un flujo o corriente de datos que fluyen entre un origen o fuente y un destino o sumidero.

- La apertura de un archivo supone establecer la conexión del programa con el dispositivo que contiene al fichero.

- Hay 3 flujos o canales siempre abiertos:
  - **stdin**: representa la entrada estándar (teclado).
  - **stdout**: representa la salida estándar (consola).
  - **stderr**: mensaje de error por pantalla.

Teclado y Consola

- Impresión en **consola**:
  - **printf**(“\n ejemplo %d”, numero);
    - Parámetros por cada variable que queramos imprimir.

- **Teclado**:
  - **scanf**(“%d”, &numero);
    - Hay que pasar la dirección.
    - scanf da bastante problemas con el buffer de teclado. No consume el retorno de carro y la siguiente lectura puede que no la haga. Se puede utilizar la función fflush(stdin) para vaciar el buffer de teclado. Mejor utilizar gets sobre una cadena y convertir a otros tipos: int, float o double.

Tratamiento de ficheros

- Puntero FILE *, definido en stdio.h
- Esquema del trabajo con ficheros:
  - Abrir fichero, indicado modo y tipo de fichero.
  - Escribir / leer datos del fichero.
  - Cerrar fichero.
- Tipos de ficheros:
  - Binarios.
  - Texto.

Apertura de un fichero

- Función: fopen(char *nombre, char *modo). Devuelve NULL en caso de que se produzca algún error, por ejemplo que vayamos a leer un fichero que no existe.

- Indicamos el nombre del fichero y el modo de apertura.

Modos de apertura

- “r” Abre para lectura
- “w” Abre para crear un nuevo archivo. Si existe se pierden sus datos.
- “a” Abre para añadir al final.
- “r+” Abre archivo que ya existe para modificar (leer / escribir).
- “w+” Crea un archivo para leer / escribir si ya existe se pierden sus datos.
- “a+” Abre un archivo para modificar (escribir / leer) al final. Si no existe es como w+.
- Estas combinarlas con t → texto y b → binario.
- Ejemplos: rt, wt, at, r+t, w+t, a+t, rb, wb, ab, r+b, w+b, a+b.

NULL y EOF

- **NULL**: Cuando una función devuelve NULL, indica que la operación no se puede realizar.

- **EOF**: Las funciones de E/S generalmente empiezan por f, si la operación falla devuelve EOF.

Ejemplo

```
#include <stdio.h>
#include <stdlib.h>
int main(void){
    FILE *f1, *f2;
    char s1[] = “ejemplo1.dat”, s2[] = “ejemplo2.dat”;
    f1 = fopen(s1, “rt”);  f2 = fopen(s2, “wb”);
    if (f1 == NULL || f2 == NULL){
        puts(“Error al abrir archivos”);
        exit (1);
    }
}
```

Cerrar un fichero

- int fclose(FILE *) Cierra el fichero especificado mediante el puntero a FILE, devuelve EOF en caso de producirse algún error.

- Ejemplo: /* A partir del anterior */

  fclose(f1);

  fclose(f2);

Para escribir en ficheros de texto

- Para carácter:
  - int putc(int, FILE *)
  - int fputc(int, FILE *): Ambas escriben un carácter en el fichero especificado, devuelve el carácter si ha ido todo bien, EOF en caso de error.
- Para cadenas:
  - int fputs(char *cad, FILE *f): Devuelve EOF si no ha podido escribir la cadena en el fichero.
- Definidas en stdio.h

Para leer de ficheros

- De carácter en carácter:
  - int getc(FILE *)
  - int fgetc(FILE *): Ambas leen un carácter del fichero y lo devuelven en caso de error devuelve EOF.
- Para cadenas de caracteres:
  - char *fgets(char *cad, int n, FILE *f): Devuelve la cadena leída en caso contrario devuelve NULL. Con n representamos el número de char a leer.

Mas funciones sobre ficheros

- int fprintf(FILE *f, const char * formato): Igual que printf pero con archivos.
- int fscanf(FILE *f, const char *formato): Igual que scanf pero con archivos.
- int feof(FILE *f): Devuelve algo <> 0 cuando se lee el fin del fichero.
- void rewind(FILE *f): Situa el puntero del archivo al inicio de este. Y se puede volver a leer.

E / S de tipos definidos por el usuario

- Para archivos binarios.

- fwrite(dirección_buffer, tamaño, num_elementos, puntero_archivo): Escribe un buffer de cualquier tamaño en un archivo binario.

- fread(direccion_buffer, tamaño, n, puntero_archivo: Lee de un archivo n bloques de bytes y los almacena en un buffer. Devuelve el nro. De bytes que lee y debe coincidir con n.

Funciones para acceso aleatorio

- El acceso aleatorio a los datos se realiza mediante la posición, es decir, el lugar relativo que ocupan.
- Tenemos que tener que en cuenta lo que ocupa cada registro.
- Funciones fseek y ftell.

Función fseek

- fseek(puntero_archivo, desplazamiento, origen): podemos tratar el archivo como si fuera un array.
  - Desplazamiento siempre se hará en función de lo que ocupa el registro. El 4 → 4 * sizeof(tipo).
  - Origen:
    - 0 → SEEK_SET, desde el inicio.
    - 1 → SEEK_CUR, desde la posición actual del puntero del archivo.
    - 2 → SEEK_END: Desde el final del archivo.

Función ftell

- long int ftell(FILE *f): Podemos obtener la posición actual del archivo y pasando como argumento el puntero al archivo.