

Tema 7: Ficheros

Oscar Perpiñán Lamigueiro

1 Introducción

2 Lectura y escritura de ficheros

3 Miscelánea

Introducción

- Hasta ahora:
 - ▶ Introducción de datos desde el **teclado**.
 - ▶ Presentación de datos en **pantalla**.
 - ▶ **Los datos se pierden** cuando finaliza el programa.
- Ahora vamos a ver:
 - ▶ **Almacenamiento de datos** en ficheros que pueden ser leídos por el programa.
 - ▶ **Operaciones con ficheros**: apertura, lectura y/o escritura, y cierre.

Tipo FILE

En C se emplea la estructura de datos de tipo FILE (declarada en `stdio.h`):

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

```
int main ()
```

```
{
```

```
    FILE *pf;
```

```
    return 0;
```

```
}
```

1 Introducción

2 Lectura y escritura de ficheros

3 Miscelánea

Abrir un fichero: fopen

fopen abre un fichero para para leer y/o escribir en él.

```
FILE *fopen (const char *nombre, const char *modo);
```

- nombre: nombre del fichero (*debe respetar las normas del sistema operativo en el que se ejecute el programa*).
- modo: indica cómo se va a abrir el fichero:
 - ▶ lectura: r
 - ▶ escritura: w
 - ▶ añadir: a
- Devuelve un puntero a una estructura de tipo FILE o un puntero nulo NULL si se ha producido un error.

Ejemplo de fopen

```
#include <stdio.h>
int main ()
{
    FILE *pf;
    // Atención a los separadores en la ruta del fichero,
    //y a las comillas dobles
    pf = fopen("c:/ejemplos/fichero.txt", "r");

    if (pf == NULL)
    {
        printf("Error al abrir el fichero.\n");
        return -1;
    }
    else
    {
        printf("Fichero abierto correctamente.\n");
        return 0;
    }
}
```

Cerrar un fichero: `fclose`

`fclose` cierra un fichero previamente abierto con `fopen`

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

- El puntero `pf`, de tipo `FILE`, apunta al fichero.
- La función devuelve 0 si el fichero se cierra correctamente o EOF si se ha producido un error.

Escritura de ficheros: fprintf

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

- Escribe en un fichero con el formato especificado (**igual que printf**)
- Devuelve el número de caracteres escritos, o un valor negativo si ocurre un error.

Ejemplo de fprintf

```
#include <stdio.h>

int main(){
    FILE *pf;
    int vals[3] = {1, 2, 3};
    // Abrimos fichero para escritura
    pf = fopen("datos.txt", "w");
    if (pf == NULL)
    {
        // Si el resultado es NULL mensaje de error
        printf("Error al abrir el fichero.\n");
        return -1;
    }
    else
    {
        // Si ha funcionado, comienza escritura
        fprintf(pf, "%i, %i, %i",
                vals[0], vals[1], vals[2]);
        fclose(pf); // Cerramos fichero
        return 0;
    }
}
```

Lectura de ficheros: fscanf

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

- Lee desde un fichero con el formato especificado (**igual que scanf**)
- Devuelve el número de argumentos que han sido leídos y asignados o EOF¹ si se detecta el final del fichero.

¹EOF (*End Of File*) es la marca de final de fichero.

Ejemplo de fscanf

```
#include <stdio.h>

int main()
{
    int i, n, vals[3];
    FILE *pf;
    // Abrimos fichero para lectura
    pf = fopen("datos.txt", "r");
    // Leemos datos separados por comas
    n = fscanf(pf, "%i, %i, %i",
               &vals[0], &vals[1], &vals[2]);
    printf("Se han leído %i argumentos.\n", n);
    fclose(pf);
    // Mostramos en pantalla lo leído
    for (i = 0; i < 3; i++)
        printf("%i\t", vals[i]);

    return(0);
}
```

Lectura de datos con separadores

Por ejemplo, para leer datos separados por punto y coma empleamos: `[^;]`

Ejemplo

Sea un fichero con el siguiente contenido:

Jorge Rodríguez; Profesor; 35; 84.4

Para leerlo:

```
fscanf(pf, "%[^;];%[^;];%d;%f\n",  
       nombre, tipo, &edad, &peso);
```

Marca de final de fichero EOF

- Cuando se crea un fichero nuevo con `fopen` se añade automáticamente al final la marca de fin de fichero EOF (*end of file*).
- Es una marca escrita al final de un fichero que indica que no hay más datos.
- Cuando se realizan operaciones de lectura o escritura es necesario comprobar si se ha alcanzado esta marca.

Comprobación de EOF

- `fEOF` detecta el final del fichero: devuelve un valor distinto de cero después de la primera operación que intente leer después de la marca final del fichero.

```
while (fEOF(pf) == 0)
{
    // Operaciones de L/E
}
```

- `fscanf` y `fprintf` devuelven `EOF` cuando alcanzan la marca. Se puede emplear directamente este resultado (sin necesidad de `fEOF`)

```
while(fscanf(...) !=EOF )
{
    // Sentencias
}
```

Ejemplo: número de líneas de un fichero

```
#include <stdio.h>

int main()
{
    int i, nLineas = 0;
    char x; // Variable auxiliar
    FILE *pf;
    pf = fopen("lorem_ipsum.txt", "r");
    // Leemos caracter a caracter
    while (fscanf(pf, "%c", &x) != EOF)
        //Si lo leído es un salto de línea
        if (x == '\n')
            //incrementamos el contador
            ++nLineas;
    printf("%i", nLineas);
    return 0;
}
```


- 1 Introducción
- 2 Lectura y escritura de ficheros
- 3 **Miscelánea**

stdin, stdout, and stderr

- Al ejecutarse un programa de C se abren tres ficheros de forma automática (identificados por tres punteros de tipo FILE):
 - ▶ stdin: entrada estándar del programa (habitualmente el teclado).
 - ▶ stdout: salida estándar del programa (habitualmente la pantalla).
 - ▶ stderr: fichero estándar de error.

Ejemplo

```
#include <stdio.h>

int main(){
    printf("hello there.\n");
    fprintf(stdout, "hello there.\n");
    return 0;
}
```

Movimiento en un fichero

fseek

```
int fseek(FILE *stream, long int offset, int whence)
```

- Desplaza a una posición en un fichero
- offset (long): valor (en bytes) a ir desde whence
- whence:

SEEK_SET Comienzo del fichero

SEEK_CUR Posición actual

SEEK_END Final del fichero

ftell

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

- Devuelve la posición actual respecto del inicio del fichero.
- Las unidades suelen ser **bytes**.
- Es una función de tipo long

Ejemplo: nº de bytes de un fichero

```
#include <stdio.h>

int main()
{
    long int fsize; // tamaño del fichero
    FILE *pf;
    pf = fopen("datos.txt", "r");
    // Desplaza al final
    fseek(pf, 0, SEEK_END);
    //Almacena la posición
    fsize = ftell(pf);
    printf("El fichero tiene %li bytes.\n",
        fsize);
    return 0;
}
```