Tema 7: Ficheros

Oscar Perpiñán Lamigueiro

Introducción

Apertura y cierre de ficheros

Escritura de ficheros

Lectura de ficheros

Utilidades

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.
  - Funciones de la biblioteca estándar de entrada/salida, `stdio.h`, que permiten manejar ficheros.

Tipo FILE

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

```
#include <stdio.h>
void main ()
{
  FILE *pf;
}
```

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.

Abrir un fichero: `fopen`

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

- Abre un fichero para para leer y/o escribir en él.
  - `nombre` es el 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`).
- La función 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;
  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;
    }
}
```

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.

Cerrar un fichero: `fclose`

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

- Cierra un fichero.
- El argumento es el puntero de tipo `FILE` que 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

```c
#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;
    }
}
```

stdout

Salida estándar del programa (habitualmente la pantalla).

```
printf("hello there.\n");
fprintf(stdout, "hello there.\n");
```

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 `EOF` si se detecta el final del fichero.

Ejemplo

```c
#include <stdio.h>

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

  return(0);
}
```

stdin

Entrada estándar del programa (habitualmente el teclado).

```
scanf("%i", &i);
fscanf(stdin, "%i", &i);
```

fgetc y fgets

- La función `fgetc` lee un carácter del fichero y lo **devuelve como** `int`:

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

- La función `fgets` lee una cadena de caracteres del fichero:

```
char *fgets (char *cadena, int n, FILE *pf);
```

Ejemplo

```c
#include <stdio.h>

int main()
{
  int i;
  char texto[27];
  FILE *pf;
  // Abrimos fichero para lectura
  pf = fopen("lorem_ipsum.txt", "r");
  // Leemos los 26 primeros caracteres
  for (i = 0; i < 26; i++)
    texto[i] = fgetc(pf);
  // Añadimos caracter nulo
  texto[26] = '\0';
  // Mostramos resultado
  printf("%s", texto);
  fclose(pf);
  return(0);
}
```

Especificador de formato [^]

- [^;]
- [^\n]

Ejemplo ¡COMPROBAR!

Fichero:

Jorge Rodríguez; Profesor; 35; 84.4
Mercedes Pérez; Estudiante; 21; 56.2

Lectura:

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

Introducción

Apertura y cierre de ficheros

Escritura de ficheros

Lectura de ficheros

Utilidades

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
}
```

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 a partir del inicio del fichero.
- Las unidades suelen ser bytes.
- Es una función de tipo `long`

Ejemplo: longitud (bytes) de un fichero

```c
#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 elementos.\n",
         fsize);
  return 0;
}
```