Tema 1. Ficheros

FUNDAMENTOS DE PROGRAMACIÓN II

Profesor: Baldomero Imbernón Tudela

Escuela Politécnica Superior
Grado en Ingeniería Informática

Contenidos

- **Entrada/salida**
  - Punteros a ficheros
  - Streams estándares
  - Ficheros de texto y ficheros binarios
- Trabajando con ficheros
  - Operaciones generales
  - E/S ficheros texto
    - Formateada
    - Carácter
    - Líneas
    - Cadenas
  - E/S ficheros binarios
    - Bloques
    - Posicionamiento en ficheros
- Ejemplos ficheros texto

Entrada/Salida

- La librería de E/S es la parte más grande e importante de la librería estándar de C.
- Hasta ahora se ha visto parte de stdio.h, concretamente funciones tales como:
  - `printf`, `scanf`, `puts`, `gets`...

... pero hay más funciones.

- Es importante destacar que la E/S a través de la librería estándar **utiliza buffers** para mejorar el rendimiento de las operaciones.

Flujos: **Streams**

- En C, el término **stream** representa cualquier fuente de entrada de datos o cualquier destino de salida de datos.
  - Muchos de los programas pequeños, como los ejemplos de la asignatura, obtienen sus datos desde un flujo de entrada, usualmente el teclado, y, escriben su salida en otro flujo, usualmente la pantalla.
  - Otros programas más grandes necesitan utilizar otros streams como por ejemplo:
    - Ficheros (en discos duros, DVD, CD, memorias USB, etc.).
    - Puertos de red.
    - Impresoras.
    - Etc.

Streams

- La librería **`stdio.h`** proporciona varias funciones que son capaces de trabajar con varios tipos de streams, no únicamente los basados en ficheros.
- El manejo correcto de streams en C requiere conocer los siguientes conceptos:
  - **Punteros a ficheros**
  - **Flujos estándar**
  - **Ficheros de texto y ficheros binarios**

Contenidos

- Entrada/salida
  - **Punteros a ficheros**
  - Streams estándares
  - Ficheros de texto y ficheros binarios
- Trabajando con ficheros
  - Operaciones generales
  - E/S ficheros texto
    - Formateada
    - Carácter
    - Líneas
    - Cadenas
  - E/S ficheros binarios
    - Bloques
    - Posicionamiento en ficheros
- Ejemplos ficheros texto

Punteros a ficheros:

(File Pointers)

- El acceso a un flujo dentro de la librería estándar de C se realiza a través de un puntero a una estructura del tipo **`FILE`**.
  - El tipo FILE se declara en la librería <stdio.h>
- Si un programa necesita acceder a dos streams basta que haga las siguientes declaraciones:

```
FILE *fp1, *fp2;
```

- Un programa puede declarar cualquier número de streams que desee,
  - El sistema operativo siempre impone un **límite de streams** que un programa puede tener abiertos en un momento dado.

Contenidos

- Entrada/salida
  - Punteros a ficheros
  - **Streams estándares**
  - Ficheros de texto y ficheros binarios
- Trabajando con ficheros
  - Operaciones generales
  - E/S ficheros texto
    - Formateada
    - Carácter
    - Líneas
    - Cadenas
  - E/S ficheros binarios
    - Bloques
    - Posicionamiento en ficheros

Streams estándares

- **`<stdio.h>`** proporciona tres streams estándares:

Nombre	Stream	Significado por defecto
stdin	Standard input	Teclado
stdout	Standard output	Pantalla
stderr	Standard error output	Pantalla

- Cualquier proceso que se ejecuta siempre tendrá acceso a esos tres streams, no se declaran, no se abren y no se cierran.

Contenidos

- Entrada/salida
  - Punteros a ficheros
  - Streams estándares
  - **Ficheros de texto y ficheros binarios**
- Trabajando con ficheros
  - Operaciones generales
  - E/S ficheros texto
    - Formateada
    - Carácter
    - Líneas
    - Cadenas
  - E/S ficheros binarios
    - Bloques
    - Posicionamiento en ficheros
- Ejemplos ficheros texto

Ficheros de texto y ficheros binarios (VI)

- `<stdio.h>` soporta dos tipos de ficheros:
  - **Ficheros de texto.**
    - Los byte de un fichero de texto representan caracteres, de tal manera que se pueden examinar y editar directamente.
    - Los caracteres deben seguir una codificación estándar (UTF-8, ISO-8859, etc.).
    - Muy usado en programación. Los programas en C se escriben en ficheros de texto.
  - **Ficheros binarios.**
    - No necesariamente representan texto.
    - Un fichero binario es un conjunto de bytes con significado para el sistema operativo (como un fichero ejecutable o una DLL) o para cualquier aplicación (un fichero de Word tiene sentido para Microsoft Word).

Ficheros de texto y ficheros binarios (VII)

- Los ficheros de texto tienen **dos características** que no poseen los ficheros binarios:

1.- Están divididos en líneas.

- Cada una de las líneas que conforman el fichero de texto terminan con uno o dos caracteres especiales.
- El **carácter de terminación** depende del sistema operativo:
  - Windows: formado por dos caracteres, un retorno de carro seguido de un salto de línea (0x0a + 0x0d).
  - UNIX y MacOS: formado por un salto de línea (0x0a).

2.- Pueden contener un carácter especial fin de fichero.

- Algunos sistemas operativos pueden utilizar un carácter especial para marcar el final del fichero.
- Muchos sistemas operativos modernos, incluyendo UNIX no lo utilizan.

Contenidos

- Entrada/salida
  - Punteros a ficheros
  - Streams estándares
  - Ficheros de texto y ficheros binarios
- **Trabajando con ficheros**
  - Operaciones generales
  - E/S ficheros texto
    - Formateada
    - Carácter
    - Líneas
    - Cadenas
  - E/S ficheros binarios
    - Bloques
    - Posicionamiento en ficheros
- Ejemplos ficheros texto

Trabajando con ficheros

- Trabajar con un fichero siempre implica tres pasos:

  1.- Abrir el fichero

  2.- Leer y/o escribir

  3.- Cerrar el fichero

Trabajando con ficheros

- Para trabajar con ficheros, la librería estándar nos proporciona un gran conjunto de funciones.
- Estas funciones las podemos clasificar en:
  - Operaciones generales sobre ficheros.
  - Entrada/salida formateada.
  - Entrada/salida de carácter.
  - Entrada/salida de líneas.
  - Entrada/salida de bloques.
  - Posicionamiento en ficheros.
  - Entrada/salida de cadenas.

Contenidos

- Entrada/salida
  - Punteros a ficheros
  - Streams estándares
  - Ficheros de texto y ficheros binarios
- Trabajando con ficheros
  - **Operaciones generales**
  - E/S ficheros texto
    - Formateada
    - Carácter
    - Líneas
    - Cadenas
  - E/S ficheros binarios
    - Bloques
    - Posicionamiento en ficheros
- Ejemplos ficheros texto

Operaciones generales: `fopen`

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

- Antes de poder utilizar un stream, es obligatorio abrirlo con la función fopen.
  - La función necesita el nombre del fichero y un modo.
  - La función devuelve un puntero a FILE. Sucesivas operaciones sobre el fichero requerirán este valor.
- El argumento path representa el nombre del fichero, puede ser una ruta absoluta o relativa.
- Podemos abrir un fichero en Windows con la llamada:

```
fp = fopen(C:\\proyecto\\fichero.dat., .r.);
```

- En UNIX, la misma llamada sería:

```
fp = fopen(./proyecto/fichero.dat., .r.);
```

Modos de `fopen`

- La forma de abrir un fichero con fopen depende de:
  - Las operaciones que se van a realizar sobre él.
  - Si el fichero es un fichero de texto o binario.
- Los **modos** para un fichero de texto son:

"r" Abre un fichero para lectura. El fichero debe existir.

"w" Crear fichero vacío para escribir. Si el fichero existe, si contenido se elimina.

"a" Abre un fichero para añadir datos al final, si no existe lo crea.

"r+" Abre un fichero para lectura y escritura, desde el principio del fichero. El fichero debe existir.

"w+" Crea un fichero vacío para leer y escribir.

"a+" Abrir un fichero para leer y añadir contenido al final.

Modos de `fopen`

- El modo puede añadir una '**b**' para indicar que se va a abrir un fichero binario.
- Los modos quedarían entonces así:

“rb” Abre un fichero para lectura.

“wb” Abre un fichero para escritura, si no existe lo crea.

“ab” Abre un fichero para añadir datos, si no existe lo crea.

“rb+” Abre un fichero para lectura y escritura, desde el principio del fichero.

“wb+” Abre un fichero para lectura y escritura, si el fichero existe se trunca todo su contenido.

“ab+” Abre un fichero para lectura y escritura (añadiendo si el fichero existe).

Operaciones generales: `fclose`

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

- Permite cerrar un fichero.
- La función devuelve:
  - Cero si el fichero se cierra correctamente.
  - Un código de error.

Operaciones generales: `remove`

```
int remove(const char *pathname);
```

- Borra un fichero.
- La función devuelve:
  - Cero si el fichero se elimina correctamente.
  - Un código de error.
- Ejemplo:

```
remove("/tmp/foto.jpg");
```

Operaciones generales: `rename`

```
int rename(const char *oldpath, const char newpath);
```

- Renombra un fichero.
- La función devuelve:
  - Cero si el fichero se elimina correctamente.
  - Un código de error.
- Si el fichero a renombrar está abierto, hay que cerrarlo antes de utilizar la función rename.

Contenidos

- Entrada/salida
  - Punteros a ficheros
  - Streams estándares
  - Ficheros de texto y ficheros binarios
- Trabajando con ficheros
  - Operaciones generales
  - **E/S ficheros texto**
    - **Formateada**
    - Carácter
    - Líneas
    - Cadenas
  - E/S ficheros binarios
    - Bloques
    - Posicionamiento en ficheros
- Ejemplos ficheros texto

Entrada/salida formateada

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

- La función fprintf funciona exactamente igual que printf pero permite realizar la salida sobre el stream definido a través del argumento FILE.
  - Ejemplo:

    ```
    printf("Total: %d\n", total);
    ```
  - ... muestra el mensaje en pantalla, mientras que:

    ```
    fprintf(fp, "Total: %d\n", total);
    ```
  - ...escribe el mensaje en el stream representado por fp.

Entrada/salida formateada

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

- La función fscanf funciona exactamente igual que scanf pero permite realizar la entrada sobre el stream definido a través del argumento `FILE`
- Ambas funciones **devuelven el número de elemento leídos**.

Entrada/salida formateada

```
void clearerr(FILE *stream);
int feof(FILE *stream);
int ferror(FILE *stream);
```

- Si pedimos leer de un stream n items, pero leemos menos, es que algo ha ido mal. Tres son las posibilidades:
  - End-of-file. La función se encontró con el final del fichero antes de completar la petición de lectura.
  - Error de lectura. La función no pudo leer caracteres del stream.
  - Error de coincidencia. El ítem leído estaba en un formato erróneo.
- **`feof`** comprueba si se ha alcanzado el final del fichero.
- **`ferror`** comprueba si se ha producido un error.
- **`clearerr`** inicializa los indicadores de error y fin de fichero.

Contenidos

- Entrada/salida
  - Punteros a ficheros
  - Streams estándares
  - Ficheros de texto y ficheros binarios
- Trabajando con ficheros
  - Operaciones generales
  - E/S ficheros texto
    - Formateada
    - **Carácter**
    - Líneas
    - Cadenas
  - E/S ficheros binarios
    - Bloques
    - Posicionamiento en ficheros
- Ejemplos ficheros texto

Entrada/salida de carácter

```
int fputc(int c, FILE *stream);
int putc(int c, FILE *stream);
int putchar(int c);
```

- **`putchar`** muestra un carácter en la salida estándar.
- **`fputc`** y **`putc`** son funciones más generales que escriben un carácter en el stream que se indique.
- La función devuelve:
  - Si ocurre algo, devuelven **`EOF`** y fijan un indicador de error.
  - Si todo va bien, devuelven el carácter mostrado/escrito.

Entrada/salida de carácter

```
int fgetc(FILE *stream);
int getc(FILE *stream);
int getchar(void);
```

- **`getchar`** lee un carácter de la entrada estándar.
- **`fgetc`** y **`getc`** son funciones más generales que leen un carácter del stream que se indique.
- La función devuelve:
  - Si ocurre algo, devuelven **`EOF`** y fijan un indicador de error.
    - Por esta razón las funciones devuelven un carácter unsigned char que se convierte a través de un casting en int.
  - Si todo va bien, devuelven el carácter mostrado/escrito.

Contenidos

- Entrada/salida
  - Punteros a ficheros
  - Streams estándares
  - Ficheros de texto y ficheros binarios
- Trabajando con ficheros
  - Operaciones generales
  - E/S ficheros texto
    - Formateada
    - Carácter
    - **Líneas**
    - Cadenas
  - E/S ficheros binarios
    - Bloques
    - Posicionamiento en ficheros
- Ejemplos ficheros texto

Entrada/salida de líneas

```
int fputs(const char *s, FILE *stream);
int puts(const char *s);
```

- Estas funciones son más adecuadas para streams de texto, pero también se pueden usar con streams binarios.
- **`puts`** muestra una línea en la salida estándar.
  - puts muestra además el retorno de carro.
- **`fputs`** es una función más general que escribe una línea en el stream que se indique.
  - fputs no escribe el retorno de carro a no ser que la cadena lo incluya.
- La función devuelve:
  - Si ocurre algo, devuelven **`EOF`**, sino un número positivo.

Entrada/salida de líneas

```
char *fgets(char *s, int size, FILE *stream);
char *gets(char *s);
```

- **`gets`** lee una línea de texto de la entrada estándar.
- **`fgets`** es una función más general que lee una línea del stream que se indique.
  - **`fgets`** lee **size - 1** caracteres a no ser que se encuentre un retorno de carro antes. Almacena los caracteres, si lee el retorno de carro, también lo almacena
  - Al final de la cadena, almacena un carácter '\0'.
- La función devuelve:
  - Si ocurre algo un puntero a NULL
  - Si todo va bien, un puntero a la cadena leída

Contenidos

- Entrada/salida
  - Punteros a ficheros
  - Streams estándares
  - Ficheros de texto y ficheros binarios
- Trabajando con ficheros
  - Operaciones generales
  - E/S ficheros texto
    - Formateada
    - Carácter
    - Líneas
    - **Cadenas**
  - E/S ficheros binarios
    - Bloques
    - Posicionamiento en ficheros
- Ejemplos ficheros texto

Entrada/salida de cadenas

```
int sprintf(char *str, const char *format, ...);
int snprintf(char *str, size_t size,
             const char *format, ...);
int sscanf(const char *str, const char *format, ...);
```

- Estas funciones no pueden manejar streams, pero tienen la habilidad de mostrar/leer desde cadenas como si éstas fuesen streams.

Contenidos

- Entrada/salida
  - Punteros a ficheros
  - Streams estándares
  - Ficheros de texto y ficheros binarios
- Trabajando con ficheros
  - Operaciones generales
  - E/S ficheros texto
    - Formateada
    - Carácter
    - Líneas
    - Cadenas
  - **E/S ficheros binarios**
    - **Bloques**
    - Posicionamiento en ficheros
- Ejemplos ficheros texto

Entrada/salida de bloques

```
size_t fread(void *direccion, int tam, int numdatos,
             FILE *stream);
```

- Lee **`tam * numdatos`** bytes del fichero, que se guardan a partir de la dirección de memoria que se indica.
- Los bytes leídos se almacenan a partir de dirección.
- Devuelve el número de datos que ha conseguido leer
  - Si ese valor es menor que **numdatos**, es porque hemos llegado al final del fichero y no se ha podido efectuar la lectura completa.

Entrada/salida de bloques

```
size_t fwrite(void * direccion, int tam, int numdatos,
              FILE * stream);
```

- Escribe en el fichero **`tam * numdatos`** bytes que se encuentran a partir de la dirección de memoria que se indica.
- Los bytes leídos se almacenan a partir de dirección.
- Devuelve el número de datos que ha conseguido escribir
  - Si devuelve un valor que es menor que **numdatos**, es porque hemos llegado al final del fichero y no se ha podido efectuar la lectura completa.

Contenidos

- Entrada/salida
  - Punteros a ficheros
  - Streams estándares
  - Ficheros de texto y ficheros binarios
- Trabajando con ficheros
  - Operaciones generales
  - E/S ficheros texto
    - Formateada
    - Carácter
    - Líneas
    - Cadenas
  - **E/S ficheros binarios**
    - Bloques
    - **Posicionamiento en ficheros**
- Ejemplos ficheros texto

Posicionamiento en ficheros

```
int fgetpos(FILE *stream, fpos_t *pos);
int fsetpos(FILE *stream, fpos_t *pos);
int fseek(FILE *stream, long offset, int whence);
long int ftell(FILE *stream);
void rewind(FILE *stream);
```

- **Cada stream tiene asociado una posición de fichero**.
  - Cuando se abre un fichero, el apuntador se coloca al principio. Si un fichero se abre para añadir, el puntero se posiciona al final.
- Estas funciones permiten manejar el apuntador a la posición actual del fichero.

Posicionamiento en ficheros

RESUMEN FUNCIONES POSICIONAMIENTO
`int fgetpos(FILE *stream, fpos_t *pos);`
Devuelve posición actual del fichero (se almacena en “pos”)
`int fsetpos(FILE *stream, fpos_t *pos);`
Establece posición actual al valor indicado en “pos”
`void rewind(FILE *stream);`
Coloca el indicador de posición de fichero al comienzo del mismo.
`int fseek(FILE *stream, long offset, int whence);`
Modifica posición del fichero al valor “whence + offset”. “whence” puede ser: SEEK_SET, SEEK_CUR, SEEK_END.
`long int ftell(FILE *stream);`
Devuelve posición actual del fichero, respecto al comienzo, en caracteres.

Contenidos

- Entrada/salida
  - Punteros a ficheros
  - Streams estándares
  - Ficheros de texto y ficheros binarios
- Trabajando con ficheros
  - Operaciones generales
  - E/S ficheros texto
    - Formateada
    - Carácter
    - Líneas
    - Cadenas
  - E/S ficheros binarios
    - Bloques
    - Posicionamiento en ficheros
- **Ejemplos ficheros texto**

Ejemplo: función para leer un entero, int, de la entrada estándar

```
int leer_int(void)
{
    int d;
    char buf[LINE_MAX];

    fgets(buf, sizeof(buf), stdin);
    sscanf(buf, "%d", &d);

    return d;
}
```

Ejemplo: copiar un fichero con getc/putc

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    FILE *fuente, *destino;
    int ch;
    if ((fuente = fopen(“fichero1.txt”, ”r")) == NULL) {
        fprintf(stderr, "No se puede abrir el fichero fuente \n”);
        exit (EXIT_FAILURE);
    }
    if ((destino = fopen(“fichero2.txt”, ”w")) == NULL) {
        fprintf(stderr, "No se puede abrir el fichero destino \n”);
        fclose(fuente);
        exit (EXIT_FAILURE);
    }
    while ((ch = fgetc(fuente)) != EOF)
        fputc(ch, destino);
    fclose(fuente);
    fclose(destino);
    return (EXIT_SUCCESS);
}
```

Ejemplo: generar un fichero de texto con un contenido

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char** argv)
{
    FILE *destino;
    int ch;
     if ((destino = fopen(“salida.txt”, "w")) == NULL) {
        fprintf(stderr, "No se puede abrir el fichero \n”);
        exit (EXIT_FAILURE);
    }
    fputs("Estamos creando un fichero de texto...\n", destino);
    fputs("con este contenido.\n", destino);
    fprintf(destino, “%d + %d = %d \n”, 3, 4, 7);
    fputs("Aquí terminamos.\n", destino);
    fclose(destino);
    return (EXIT_SUCCESS);
}
```

Bibliografía

- King, K.N. **C Programming. A modern approach.** 2ªed. Ed. W.W. Norton & Company. Inc. 2008. Chapter 22.
- Khamtane Ashok**. Programming in C**. Ed. Pearson. 2012.
- Ferraris Llanos, R. D. **Fundamentos de la Informática y Programación en C**. Ed. Paraninfo. 2010.