Operaciones de E/S en ANSI C

Las operaciones de entrada/salida estándar (realizadas habitualmente con `printf` y `scanf`) se realizan en realidad sobre ficheros que representan los dispositivos mediante los cuales el usuario interacciona con el ordenador.

La biblioteca estándar **stdio** (incluida en el ANSI C) nos permite manipular ficheros desde nuestros programas en C, para lo cual hemos de incluir la siguiente directiva al comienzo de nuestros ficheros de código (para evitar que el compilador encuentre tipos y funciones no definidas):

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

E/S de caracteres

Al nivel más bajo, las operaciones de entrada/salida se realizan con bytes, si bien resulta engorroso tener que programar a este nivel:

getchar
La función `getchar` devuelve el siguiente carácter leído desde el teclado.

putchar
Muestra un carácter a través del dispositivo de salida estándar
(la pantalla, generalmente).

> NOTA: La mayoría de los sistemas operativos utilizan buffers para gestionar las operaciones de entrada/salida, por lo que no se leerá un carácter hasta que el usuario introduzca un retorno de carro y, posiblemente, no se mostrará nada por pantalla hasta que se tenga una línea completa por mostrar.

Ejemplo: Contar el número de caracteres de un fichero

```
#include <stdio.h>

int main ()
{
  int c = 0;

  while (getchar() != EOF)
        c ++;

  printf("%d\n", c);

  return 0;
}
```

Ejemplo: Poner un texto en mayúsculas

```
#include <ctype.h>  /* Definición de toupper */
#include <stdio.h>  /* getchar, putchar, EOF */

int main ()
{
  int ch;

  while ((ch = getchar()) != EOF)
        putchar(toupper(ch));

  return 0;
}
```

NOTA: Para indicar el final de un fichero cuando se están introduciendo los datos desde el teclado hay que escribir CONTROL+Z en Windows (CONTROL+D en UNIX/Linux).

E/S con formato

La biblioteca estándar de C incluye dos funciones que nos facilitan un control preciso sobre las operaciones de entrada y salida:

printf
Nos permite especificar el formato en que queremos mostrar datos por pantalla

Parámetros: Cadena de formato (cómo se visualizan los datos)
Lista de valores (datos que se visualizan)

scanf
Permite leer datos con formato desde el teclado.

Parámetros: Cadena de formato (cómo se introducen los datos)
Lista de punteros (dónde se almacenan los datos)

Cadena de formato
`%[flag][ancho][.prec][modificador][tipo]`

Tipo:

`%c`	Carácter (char)
`%s`	Cadena de caracteres (string)
`%d`	Número entero (decimal)
`%x`	Número entero (en hexadecimal)
`%f`	Número real (float)

Modificador:

`%ld`	Número entero (long int)
`%lf`	Número real (double)
`%Lf`	Número real (long double)

`[ancho]` indica el número mínimo de caracteres que se utilizarán para visualizar el dato (el espacio sobrante se rellena con espacios o con ceros).

`[.prec]` indica el número máximo de dígitos/caracteres que se mostrarán. Las cadenas de caracteres se truncan, los números reales se redondean utilizando `prec` decimales.

`[flag]`: Por defecto, los datos se justifican a la derecha. Poniendo -, los datos se justifican a la izquierda. Con + se indica que siempre queremos visualizar el signo de un número (+ ó -).

E/S de líneas

Si no estamos interesados en el formato de nuestros datos (o, simplemente, no podemos predecir su formato de antemano), podemos leer y escribir líneas completas de caracteres con otras dos funciones estándar:

gets (cadena)
Lee una línea completa
(hasta que se alcanza un retorno de carro [\n] o el final del fichero [EOF]).

NOTAS:
- `gets` devuelve NULL cuando se llega al final de la entrada.
- `gets` no comprueba la longitud de la cadena de entrada, por lo que siempre utilizaremos la función `fgets`, que sí lo hace.

puts (cadena)
Escribe una línea de texto y le añade un retorno de carro al final.

Ejemplo: Doble espaciado

```
#include <stdio.h>

int main ()
{
  char line[256]; /* ¿Suficientemente grande? */


  while ( gets(line) != NULL) {

        puts(line);

        printf("\n");
  }

  return 0;
}
```

NOTA: Se pueden mezclar libremente las operaciones de E/S con formato, E/S de caracteres y E/S de líneas.

Redirección de ficheros en Windows y UNIX

La biblioteca estándar de E/S incluye variantes de las funciones de E/S ya vistas para trabajar con ficheros, si bien podemos utilizar las funciones de E/S estándar para trabajar con ficheros si utilizamos redireccionamientos.

Los sistemas operativos más comunes permiten utilizar un fichero como entrada estándar a un programa (en vez de tener que teclear los datos a mano) y almacenar la salida estándar en un fichero de texto. Esto se puede conseguir fácilmente desde la línea de comandos del sistema operativo:

```
programa <entrada.txt
```

ejecuta el programa leyendo los datos del fichero de texto `entrada.txt`

```
programa >salida.txt
```

ejecuta el programa y guarda los resultados en el fichero `salida.txt`

```
programa <entrada.txt >salida.txt
```

lee los datos de `entrada.txt` y guarda los resultados en `salida.txt`

La redirección resulta fácil de utilizar y nos permite que un único programa funcione leyendo datos desde el teclado o desde un fichero:

> Podemos escribir los programas para que lean los datos desde el teclado y muestren los resultados por pantalla. Después los utilizaremos con datos almacenados en ficheros (y guardar en ficheros los resultados que obtengamos).

Algunos programas necesitan acceder a distintos ficheros, para lo cual recurriremos a las funciones de manipulación de ficheros definidas en la biblioteca estándar de C…

Manipulación de ficheros en C

La biblioteca `<stdio.h>` incluye un tipo de dato que nos permitirá manipular ficheros mediante punteros: el tipo `FILE *`

Para utilizar un fichero, hemos de declararlo como una variable más:

```
FILE *fichero;
```

Antes de poder acceder a él, hay que abrir el fichero. La función `fopen` devuelve el puntero a través del cual accederemos al fichero:

```
fichero = fopen(“salida.txt”, “w”)
```

Si, por cualquier motivo, el fichero no puede abrirse, la función `fopen` devuelve NULL, por lo que siempre deberemos comprobar el resultado de llamar a la función `fopen`:

```
if (fichero == NULL)
   fprintf (stderr, “No se pudo acceder al fichero\n”);
```

Siempre que se abre un fichero, hay que cerrarlo después de usarlo:

```
fclose (fichero);
```

Suele existir un límite sobre el número de ficheros que pueden estar abiertos simultáneamente, por lo que es conveniente cerrar el fichero justo después de utilizarlo si no se va a volver a acceder a él.

Existen tres ficheros predefinidos en la biblioteca estándar de C:

- ✓ `stdin` (entrada estándar: el teclado o un fichero redirigido).
- ✓ `stdout` (salida estándar: la pantalla o un fichero al que se envía la salida).
- ✓ `stderr` (salida estándar para mensajes de error: la pantalla).

fopen

Abre un fichero

Parámetros: Nombre del fichero
Modo de acceso

- `"r"` Lectura
- `"w"` Escritura (crear un fichero nuevo)
- `"a"` Escritura (añadir al final del fichero)

fclose

Cierra un fichero

Parámetro: Puntero al fichero (que debe estar abierto).

Operaciones de E/S con ficheros

Existen funciones para trabajar con ficheros que son análogas a las funciones de entrada/salida estándar (sólo tenemos que indicar el fichero sobre el cual deseamos realizar la operación correspondiente):

	equivale a
`putc(c,stdout)`	`putchar(c)`
`getc(stdin)`	`getchar()`
`fprintf(stdout,…)`	`printf(…)`
`fscanf(stdin,…)`	`scanf(…)`
`fputs(cadena,stdout)`	`puts(cadena)`
`fgets(cadena,longitud,stdin)`	`gets(cadena)`

Operaciones con cadenas de caracteres

sprintf/scanf

La biblioteca de E/S estándar incluye dos funciones análogas a `printf/scanf` que nos permiten trabajar directamente sobre cadenas de caracteres:

- ✓ `sprintf` almacena datos con formato en una cadena de caracteres (que debe ser lo suficientemente grande como para albergar los datos con formato).

- ✓ `sscanf` toma los datos de una cadena y los almacena en las variables especificadas de acuerdo con la cadena de formato que indiquemos (p.ej. para convertir una cadena en un valor numérico).

string.h

El estándar ANSI C incluye otra biblioteca que incluye varias funciones para facilitarnos la manipulación de cadenas de caracteres (algo muy habitual cuando trabajamos con ficheros):

```
#include <string.h>
```

Entre las funciones más utilizadas de esta biblioteca se encuentran las siguientes:

- ✓ `strcpy(dest,src)` copia la cadena `src` en `dest`, si bien tiene el mismo defecto que la función `gets`.

- ✓ `strncpy(dest,n,src)` copia la cadena `src` en `dest`, si bien tiene en cuenta el tamaño de dest. El parámetro n debe ser igual al tamaño del vector `dest` menos 1 (si queremos dejar espacio para el '\0' final).

- ✓ `strcmp(cad1,cad2)` compara las cadenas `cad1` y `cad2`. Devuelve 0 si las cadenas son iguales.

- ✓ `strlen(cadena)` devuelve la longitud de la cadena (el número de caracteres que hay hasta que nos encontramos un '\0').