**Capítulos:** 7.1, 7.2.

## 7.1 Entrada y Salida estándar

El mecanismo de entrada más simple es leer un carácter a la vez de la entrada estándar, normalmente el teclado, con **getchar**:



**getchar** regresa el siguiente carácter de la entrada cada vez que se invoca, o EOF cuanto encuentra un fin de archivo.

La constante simbólica EOF está definida en <stdio.h>. El valor es típicamente -1, pero las pruebas se deben escribir en función de EOF, de modo que sean independientes del valor específico.

Redireccionamiento de entrada: **<**. Si un programa **prog** usa **getchar**, entonces la línea de órdenes



provoca que **prog** lea los caracteres de **archent**. El cambio de la entrada se realiza de tal manera que prog mismo es ajeno al cambio. La cadena “archent” no está incluida entre los argumentos de la línea de órdenes de **argv**.

El cambio de la entrada también es invisible si la entrada viene de otro programa vía un mecanismo de interconexión (pipe): en algunos sistemas, la línea de órdenes



ejecuta tanto al programa **otroprog** como a **prog**, e interconecta la salida estándar de otroprog con la entrada estándar de prog.



**putchar(c)** se emplea para salida; coloca el carácter c en la salida estándar, que por omisión es la pantalla, putchar regresa el carácter escrito, o EOF si ocurre algún error.

La salida puede ser dirigida hacia algún archivo con **>nombrearch**. Si prog utiliza putchar,



escribirá la salida estándar hacia archsal.

Se permite la interconexión,



deja la salida estándar de prog en la entrada estándar de otroprog.

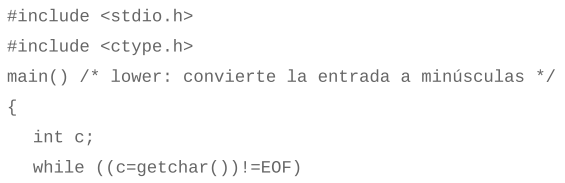
La salida producida con **printf** también encuentra su camino hacia la salida estándar. Las llamadas a **putchar** y **printf** pueden estar traslapadas (La salida aparece en el orden en que se hicieron las llamadas).

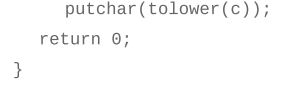
Cada archivo fuente que se refiera a una función de biblioteca de entrada/salida debe contener la línea

antes de la primera referencia.

Cuando un nombre se delimita por **< >** se realiza una búsqueda del header en algunos lugares estándar.

Por ejemplo, considérese el programa **lower**, que convierte su entrada a minúsculas





La función **tolower** está definida en <ctype.h>; convierte una letra mayúscula a minúscula, y regresa los otros caracteres intactos. Este tipo de funciones son a menudo macros, para así evitar la sobrecarga de llamados a funciones.

## 7.2 Salida con formato -- printf

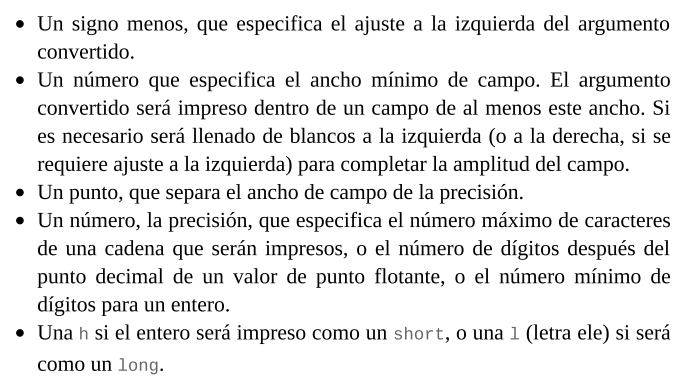


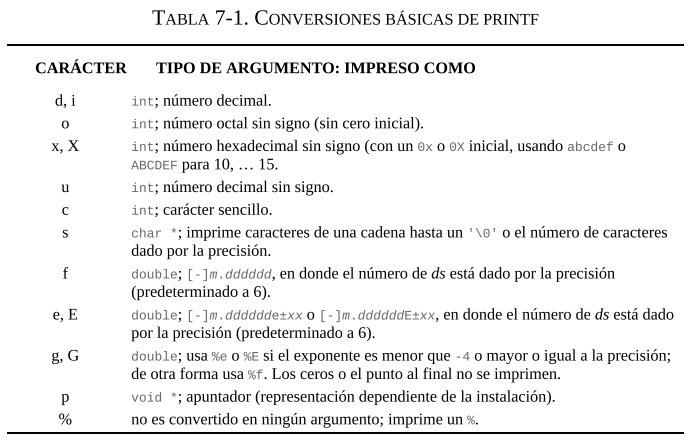
**printf** convierte, da formato e imprime sus argumentos en la salida estándar bajo el control de **format**. Regresa el número de caracteres impresos.

La cadena de formato contiene dos tipos de objetos:

Caracteres ordinarios, que son copiados al flujo de salida.

Especificaciones de conversión, cada uno de los cuales causa la conversión e impresión de los siguientes argumentos sucesivos de printf. Cada especificación de conversión comienza con un **%** y termina con un carácter de conversión. Entre el % y el carácter de conversión pueden estar, en orden:

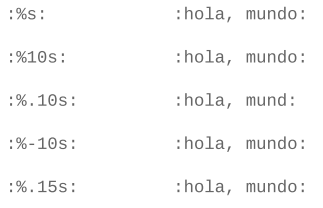


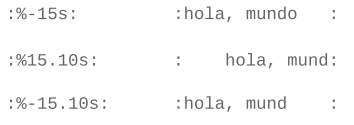


Una amplitud o precisión se puede especificar por **\***, en cuyo caso el valor se calcula convirtiendo el siguiente argumento (int). Por ejemplo, para imprimir al menos max caracteres de una cadena s,



Ejemplos:





**Advertencia:** printf emplea su primer argumento para decidir cuántos argumentos le siguen y cuáles son sus tipos. Se obtendrán resultados erróneos si no hay suficientes argumentos o si tienen tipos incorrectos.

También se debe advertir la diferencia entre estas dos llamadas:



**sprintf** realiza las mismas conversiones que printf, pero almacena la salida en una cadena:



Da formato a los argumentos que están en arg1, arg2, etc., de acuerdo con format como antes, pero coloca el resultado en **cadena** en vez de la salida estándar. **cadena** debe ser lo suficientemente grande como para recibir el resultado.