Los secretos de "printf" *

Profesor Don Colton

Brigham Young Universidad Hawaii

`printf` es la función del lenguaje C para hacer impresiones con formato. La misma función también se encuentra disponible en PERL. Éste trabajo explica cómo funciona `printf`, y cómo diseñar la apropiada especificación de formato para cualquier ocasión.

1. Introducción

En los primeros días, los programadores de computadoras escribían sus propias subrutinas para leer e imprimir números. En realidad, no es terriblemente dificultoso. Simplemente se debe alocar un arreglo de caracteres para almacenar el resultado, dividir el número por diez, guardar el resto, añadirle x30, y almacenarlo al final del arreglo. Repetir el procedimiento hasta obtener todos los dígitos. Después imprimirlo. Demasiado fácil ¿no?

Aunque fue fácil (para Einstein), igualmente requirió cierto esfuerzo. ¿Y que hay de las validaciones? ¿y números negativos? Entonces los programadores de computadoras crearon bibliotecas de funciones pregrabadas. Y fue bueno. Eventualmente, las funciones más populares de éstas fueron canonizadas a la pertenencia de las bibliotecas "estándar". La impresión de números fue suficientemente popular como para ganar este santo honor.

Ésto significó que los programadores no tuvieron que reinventar la subrutina de impresión de números una y otra vez. También significó que las opciones favoritas de todos tratasen de ser incluidas en el estándar.

Así nació `printf`.

*Título original: *Secrets of "printf"*, traducido por Patricio Moreno.

2. Impresiones sencillas

En el caso más simple, `printf` recibe un argumento: una cadena de caracteres a ser impresa. Esta cadena está compuesta de caracteres, cada uno es impreso exactamente como aparece. Entonces `printf("xyz");` simplemente imprimiría una x, luego una y, y finalmente una z. Ésto no es exactamente impresión "con formato", pero sigue siendo la base de lo que hace `printf`.

2.1. Caracteres especiales por naturaleza

Para identificar el comienzo de una cadena, se ponen comillas dobles (`"`) al comienzo. Para identificar el final de una cadena se ponen otras comillas dobles al final. Pero ¿y si en realidad se quiere imprimir unas comillas dobles? No se puede poner exactamente unas comillas dobles en el medio de una cadena porque sería malinterpretada como el marcador de fin de cadena. Las comillas dobles son un carácter especial. Las reglas normales de imprima-lo-que-ve no aplican.

Diferentes lenguajes tienen diferentes acercamientos a éste problema. Algunos requieren que el carácter especial sea ingresado dos veces. C usa barras invertidas (*backslash*, \) como carácter de escape para cambiar el significado del siguiente carácter. En consecuencia, para imprimir las comillas dobles, se ingresa barra invertida comilla doble (`\"`). Para imprimir una barra invertida, se la debe escapear anteponiéndole otra barra invertida. La primera barra invertida significa "darle al siguiente carácter su significado alternativo". La segunda tiene el significado alternativo de "imprimir una barra invertida".

Sin la barra invertida, los caracteres especiales tienen un significado naturalmente especial. Con la barra invertida, se los imprime como aparecen. La siguiente es una lista parcial:

\	escapar el siguiente caracter
\\	imprimir una barra invertida
"	comienzo o fin de cadena
\"	imprimir comillas dobles
'	comienzo o fin de una constante de caracter
\'	imprimir comilla simple
%	comenzar especificación de formato
\%	imprimir un signo de porcentaje

2.2. Caracteres especiales alternativos

Por otro lado están los caracteres que normalmente se imprimen como uno espera, pero cuando se añade la barra invertida, es entonces cuando se vuelven especiales. Un ejemplo es el carácter de nueva línea. Para imprimir una `n`, simplemente se escribe una `n`. Para imprimir una nueva línea, se escribe `\n`, invocando el significado alternativo de `n`, que es nueva línea. Aquí una lista parcial.

\a	alerta auditiva (campana -bell-)
\b	retroceso
\f	form feed
\n	nueva línea
\r	retorno de carro
\t	tabulación
\v	tabulación vertical

3. Especificaciones de formato

El poder real de `printf` se da cuando se imprime el contenido de variables. Tomemos el especificador de formato `%d` por ejemplo. Ésto imprime un número. Entonces, un número debe ser provisto para que sea impreso. Ésto se hace añadiendo otro argumento a la sentencia de `printf`, como se muestra aquí.

```
int age;
age = 25;
printf ( "I am %d years old\n", age );
```

En el ejemplo, `printf` tiene dos argumentos. El primero es una cadena: `"I am %d years old\n"`. El segundo es un entero, `age`.

3.1. La lista de argumentos

Cuando `printf` procesa sus argumentos, comienza imprimiendo los caracteres que encuentra en el primer argumento, uno por uno. Cuando encuentra un signo de porcentaje sabe que tiene una especificación de formato. Va al siguiente argumento y usa su valor, imprimiéndolo de acuerdo a esa especificación de formato. Luego vuelve a imprimir los caracteres de a uno (tomados del primer argumento).

Es correcto incluir más de una especificación de formato en la cadena de `printf`. En ese caso, el primer especificador de formato se corresponde con el primer argumento adicional, el segundo con el segundo, y así. He aquí un ejemplo:

```
int x = 5, y = 10;
printf ( "x is %d and y is %d\n", x, y );
```

3.2. Porcentaje

Toda especificación de formato comienza con un signo de porcentaje y finaliza con una letra. Las letras son escogidas para tener un significado mnemónico. Aquí una lista parcial:

%c	imprimir un carácter
%d	un número decimal (base 10)
%e	un N.º de punto flotante exponencial
%f	un N.º de punto flotante
%g	un N.º de punto flotante en formato general
%i	un entero en base 10
%o	un N.º en octal (base 8)
%s	una cadena de caracteres
%u	un N.º decimal no signado (base 10)
%x	un N.º en hexadecimal (base 16)
%%	un signo porcentaje (también funciona \%)

Para imprimir un número de manera sencilla, el especificador de formato es simplemente `%d`. He aquí algunos casos y resultados.

printf	imprime
("%d",0)	0
("%d",-7)	-7
("%d",1560133635)	1560133635
("%d",-2035065302)	-2035065302

Notar que en manera sencilla, %d, no hay un tamaño determinado para el resultado. printf simplemente toma el espacio que necesita.

3.3. La opción para el *ancho*

Como se mencionó arriba, la simple impresión de números no era suficiente. Eran deseables otras opciones especiales. La más importante era probablemente la opción para el ancho. Poniendo %5d estaba garantizado que el número ocuparа cinco espacios (más si fuese necesario, nunca menos). Esto resultaba muy útil al imprimir tablas porque números pequeños y grandes ocupaban el mismo espacio. Casi todas las impresiones eran monoespaciadas en esos días, lo que significa que una w y una i tienen el mismo espacio. Ésto, actualmente, es común en los editores de texto utilizados por programadores.

Para imprimir un número con un cierto ancho (mínimo), digamos un ancho de 5 espacios, el especificador de formato es %5d. A continuación, algunos ejemplos. (El símbolo ␣ indica explícitamente los espacios).

printf	imprime
("%5d",0)	␣␣␣␣0
("%5d",-7)	␣␣␣-7
("%5d",1560133635)	1560133635
("%5d",-2035065302)	-2035065302

Notar que para números cortos, el resultado es alineado con espacios a izquierda. Para números excesivamente largos, no hay relleno y el número completo es impreso.

Durante el uso normal, se haría que el ancho del campo sea lo suficientemente grande como para el número más largo esperable. Si los números son usualmente de uno, dos o tres dígitos, entonces %3d probablemente es adecuado. En casos anormales, se podría terminar imprimiendo un número que es demasiado largo para el campo. printf toma la decisión de imprimir completamente esos números, a pesar de que tomen demasiado espacio. Ésto es porque es preferible imprimir la respuesta correcta y que quede feo, a imprimir una respuesta incorrecta y que se vea bien.

3.4. Completando espacios

Al imprimir números pequeños como 27 en un campo con %5d, surge la pregunta sobre dónde poner el 27 y qué poner en los otros tres espacios. Podría ser impreso en los primeros dos espacios, los últimos dos o en los espacios del medio (si es que se puede determinar). Los espacios en blanco podrían ser completados con el carácter espacio, o con asteriscos (***27 o 27*** o **27*), o signos monetarios ($$$27), o signos igual (===27), o con ceros a izquierda (00027).

Estos caracteres extra a veces son llamados caracteres de "protección de cheques" porque se utilizan para evitar que gente mala cambie el monto en un cheque impreso. Es relativamente fácil cambiar un espacio por cualquier otra cosa. Es más difícil cambiar un asterisco u otro signo.

printf provee el rellenado con espacios (a izquierda o derecha) y con ceros (sólo a izquierda). Si se quiere "protección de cheques" o centrado, hay que hacer otras modificaciones. Pero incluso sin "protección de cheques" o centrado, printf tiene una colección de opciones impresionante (y desconcertante).

3.5. La opción para justificar

Al utilizar printf, los números pueden ser justificados a izquierda (impresos en el lado izquierdo del campo) o justificados a derecha (impresos en el lado derecho del campo). El modo más natural para imprimir los números parece ser justificados a derecha anteponiendo espacios. Eso es lo que %5d significa: imprimir un número en base 10 en un campo de ancho 5, con los números alineados a derecha y rellenando con espacios al principio.

Para alinear un número a la izquierda, se agrega un signo menos al especificador de formato. Para imprimir un número con un ancho de 5 espacios y justificado a izquierda el especificador de formato es %-5d.

A continuación, unos ejemplos.

printf	imprime
("%-5d",0)	0␣␣␣␣
("%-5d",-7)	-7␣␣␣
("%-5d",1560133635)	1560133635
("%-5d",-2035065302)	-2035065302

Como antes, para números cortos, el resultado es completado con espacios. Para números más largos no hay relleno, y el número no es recortado.

3.6. La opción para rellenar con ceros

Para hacer que las cosas se alineen y queden bien, es común imprimir las fechas anteponiendo ceros. Se puede escribir 25 de mayo de 1810 como 25/05/1810. También se podría escribir como 1810.05.25[1]. Notar que en ambos casos el cero no cambia el significado. Está únicamente para hacer que alineen bien en listas.

Cuando un número es rellenado con ceros, los ceros siempre van al comienzo, y el número resultante está justificado a izquierda y a derecha. En este caso el signo menos no tiene ningún efecto. Para imprimir un número con un ancho de 5 espacios y rellenado con ceros el especificador de formato es `%05d`. A continuación unos ejemplos.

printf	imprime
("%05d",0)	00000
("%05d",-7)	-0007
("%05d",1560133635)	1560133635
("%05d",-2035065302)	-2035065302

Los números cortos son rellenados con ceros. Los números largos no cambian.

3.7. Jugando con el signo más

Los números negativos siempre se imprimen con el signo menos. Los números positivos y el cero usualmente no se imprimen con un signo, pero se puede pedir uno. Un más (+) en el especificador de formato hace ese pedido.

Para imprimir un número con un ancho de 5 espacios y con su signo el especificador de formato es `%+5d`. Ejemplos:

printf	imprime
("%+5d",0)	␣␣␣+0
("%+5d",-7)	␣␣␣-7
("%+5d",1560133635)	+1560133635
("%+5d",-2035065302)	-2035065302

Notar que el cero es tratado como un número positivo. Números cortos rellenados. Números largos sin cambio.

Los signos más y menos no están relacionados. Ambos pueden aparecer en el especificador de formato.

3.8. El signo + invisible

Ésta es un poco extraña. Es un signo más invisible. En lugar de imprimir los números positivos (y el cero) con un signo más, imprime un espacio en su lugar. Ésto puede ser útil al imprimir números justificados a izquierda en los que se quiere el signo menos se destaque. Noten estas dos alternativas.

printf	imprime
("%+-5d",0)	+0␣␣␣
("%+-5d",-7)	-7␣␣␣
("%+-5d",1560133635)	+1560133635
("%+-5d",-2035065302)	-2035065302

printf	imprime
("%␣-5d",0)	␣0␣␣␣
("%␣-5d",-7)	-7␣␣␣
("%␣-5d",1560133635)	␣1560133635
("%␣-5d",-2035065302)	-2035065302

Como se mostró arriba, con el especificador de formato `%-5d` se obtenían los siguientes resultados (se muestran de nuevo para facilitar la comparación)

Notar que el signo más desaparece, pero aún se utiliza un espacio adelante del número.

[1] N. del T.: el texto original utiliza el formato de Estados Unidos de Norteamérica y la fecha 05/05/2003 ¿cuál es el mes y cuál el día?

printf	imprime
("%5d",0)	0␣␣␣␣
("%5d",-7)	-7␣␣␣
("%5d",1560133635)	1560133635
("%5d",-2035065302)	-2035065302

Notar también que se pueden combinar varias opciones en el mismo especificador de formato. En este caso, se combinaron las opciones más, menos y cinco, o espacio, menos y cinco, o sólo menos y cinco.

3.9. Más, Espacio y Cero

Aquí hay otro ejemplo sobre la combinación de varias opciones al mismo tiempo.

Usando el especificador de formato %␣05d o %0␣5d se obtienen los siguientes resultados.

printf	imprime
("% 05d",0)	␣0000
("% 05d",-7)	-0007
("% 05d",1560133635)	␣1560133635
("% 05d",-2035065302)	-2035065302

Usando %+05d o %0+5d se obtiene:

printf	imprime
("%+05d",0)	+0000
("%+05d",-7)	-0007
("%+05d",1560133635)	+1560133635
("%+05d",-2035065302)	-2035065302

Cuando se combina el más y el espacio al mismo tiempo, el espacio reserva lugar para un signo y el más lo utiliza. Es como si el espacio no fuese especificado. El más tiene prioridad sobre el espacio.

3.10. Resumen

A las opciones también se las llama "banderas" (*flags* en inglés) y entre ellas pueden aparecer en cualquier orden. Aquí, una lista parcial.

Después de las opciones, si hay, se puede especificar el mínimo ancho de campo.

flag	efecto
ninguna	alinea a derecha, con espacios
-	justifica a izquierda
0	rellena con ceros
+	con un signo más en números positivos
␣	signo más invisible

4. Imprimiendo cadenas

La opción %s permite la impresión de una cadena dentro de otra. Ejemplo:

```
char * grade;
if ( year == 11 ) grade == "junior";
printf ( "%s is a %s\n", "Fred", grade );
```

La bandera de justificación a izquierda también aplica a cadenas, pero por supuesto que el rellenar con ceros, el signo más y el signo más invisible no tienen sentido.

printf	imprime
("%5s","")	␣␣␣␣␣
("%5s","a")	␣␣␣␣a
("%5s","ab")	␣␣␣ab
("%5s","abcdefg")	abcdefg

printf	imprime
("%-5s","")	␣␣␣␣␣
("%-5s","a")	a␣␣␣␣
("%-5s","ab")	ab␣␣␣
("%-5s","abcdefg")	abcdefg

5. Punto Flotante

Los números de punto flotante son aquellos de la pinta 3.1415 que tienen un punto decimal en algún lado. Ésto es en contraste con los enteros ordinarios como 27, que no tiene punto decimal.

Las mismas banderas y reglas que aplican para enteros aplican también para números de punto flotante, pero hay algunas opciones nuevas. La más importante es la manera en que se especifica cuántos dígitos

aparecen después del punto decimal. Este número se llama la **precisión** del número.

Para el comercio común, los precios se mencionan como pesos enteros o pesos y centavos (cero o dos dígitos de precisión). Para la nafta, los precios se dan en pesos, centavos y décimas de centavos (tres dígitos de precisión)[2]. A continuación algunos ejemplos de cómo imprimir este tipo de números. Sea `e␣=␣2.718281828`.

printf	imprime
`("%.0f",e)`	`3`
`("%.0f.",e)`	`3.`
`("%.1f",e)`	`2.7`
`("%.2f",e)`	`2.72`
`("%.6f",e)`	`2.718282`
`("%f",e)`	`2.718282`
`("%.7f",e)`	`2.7182818`

Notar que si se especifica un punto y un número, el número (la precisión) indica cuántos espacios deben ser mostrados después del punto decimal.

Notar que si no se especifica el punto y la precisión en `%f`, el valor por omisión es `%.6f` (seis dígitos después del punto decimal).

Notar que si se especifica una precisión igual a cero, el punto decimal también desaparece. Si se lo quiere de todos modos, se lo debe agregar por separado (después del especificador `%f`).

Se pueden especificar el ancho como la precisión al mismo tiempo. Notar especialmente que `5.2` significa un ancho total de cinco, con dos dígitos después del punto decimal. **Es muy común y natural pensar que significa cinco dígitos antes del punto decimal y dos dígitos después, pero eso no es correcto.** Tener cuidado.

printf	imprime
`("%5.0f",e)`	`␣␣␣␣3`
`("%5.0f.",e)`	`␣␣␣␣3.`
`("%5.1f",e)`	`␣␣2.7`
`("%5.2f",e)`	`␣2.72`
`("%.7f",e)`	`2.7182818`

[2]N. del T.: aquí también se utiliza la moneda y lenguaje local

También se puede combinar la precisión con las banderas que se mencionaron antes, para especificar justificación a izquierda, rellenar con ceros, signo más, etc.

printf	imprime
`("%5.1f",e)`	`␣␣2.7`
`("%-5.1f",e)`	`2.7␣␣`
`("%+5.1f",e)`	`␣+2.7`
`("%+-5.1f",e)`	`+2.7␣`
`("%05.1f",e)`	`002.7`
`("%+05.1f",e)`	`+02.7`
`("%␣05.1f",e)`	`␣02.7`
`("%-␣5.1f",e)`	`␣2.7␣`