# Los secretos del “printf”

Professor Don Colton Brigham
Young University Hawaii

Printf es la función del lenguaje C para obtener impresiones formateadas. La misma función se encuentra también disponible en PERL. Este artículo, explica como funciona printf y como diseñar las oportunas especificaciones de formato para cada ocasión.

## 1 Antecedentes

En los primeros días, los programadores escribían sus propias subrutinas para leer e imprimir números. No es terriblemente difícil, la verdad. Simplemente se asigna una matriz de caracteres para almacenar el resultado, divida el número entre diez, conserve el resto, agréguele x30 y almacénelo al final de la matriz. Repita el proceso hasta que se encuentren todos los dígitos.

Luego imprímalo. Demasiado fácil, ¿verdad? Pero a pesar de que fue fácil (para Einstein), aún requirió un poco de esfuerzo. ¿Y qué hay de la verificación de errores y los números negativos? Entonces, los programadores crearon bibliotecas de funciones pregrabadas. Y estuvo bien. Finalmente, las funciones más populares se normalizaron para formar parte de las bibliotecas "estándar". La impresión de números fue lo suficientemente popular como para ganar este honor sagrado. Esto significaba que los programadores no tenían que reinventar la subrutina de impresión de números una y otra vez. También significaba que las opciones favoritas de todos trataban de llegar al estándar. Así nació ***printf***

## 2 Impresión sencilla

En el caso más simple, printf toma un argumento: una cadena de caracteres para imprimir. Esta cadena está compuesta de caracteres, cada uno de los cuales se imprime exactamente como aparece. Entonces printf (*“xyz”*); simplemente imprimiría una x, luego una y, y finalmente una z. Esta no es exactamente una impresión "***formateada***", pero sigue siendo la base de lo que hace printf.

### *2.1 Por supuesto caracteres especiales*

Para identificar el comienzo de la cadena, ponemos unas dobles comillas *(“)* antes de comenzar. Para identificar el final de la cadena, ponemos este mismo símbolo al final. Pero, ¿qué sucede si realmente queremos imprimir unas dobles comillas? No podemos poner exactamente una dobles comillas doble en el medio de la cadena porque se confundiría con el marcador de fin de cadena. Por tanto, las dobles comillas son un carácter especial.

Las reglas normales de imprimir lo que ves no se aplican. Diferentes lenguajes toman diferentes enfoques para este problema. Algunos requieren que se ingrese el carácter especial dos veces. C usa la barra invertida (*vírgula,* \) como carácter de escape para cambiar el significado del siguiente carácter después de él.

Por lo tanto, para imprimir unas comillas dobles, escriba tras la barra invertida unas comillas dobles. Para imprimir una barra invertida, debe escapar escribiendo otra barra invertida frente a ella. La primera barra diagonal inversa significa "dar al siguiente carácter su significado alternativo". La segunda barra diagonal inversa tiene un significado alternativo de "imprimir una barra diagonal inversa". Sin una barra diagonal inversa, los caracteres especiales tienen un significado especial natural. Con una barra invertida se imprimen a medida que aparecen. Aquí tienes una lista parcial.

| | |
|---|---|
| \<br>\\ | Carácter de escape<br>Imprime una barra invertida |
| “<br>\” | Principio o final de una cadena de texto<br>Imprime unas dobles comillas |
| ‘<br>\’ | Principio o final de una constante de caracteres<br>Imprime un apostrofe |
| %<br>\% | Comienzo de una especificación de formato<br>Impresión de un símbolo de porcentaje |

## 2.2 Caracteres especiales alternativos

Por otro lado, tenemos caracteres que normalmente se imprimen como cabría esperar, pero cuando agregamos una barra diagonal inversa, se vuelven especiales. Un ejemplo es el carácter de nueva línea. Para imprimir ann, simplemente escribimos en el documento ann. Para imprimir una nueva línea, escribimos los caracteres \n, invocando así el significado alternativo de n, que es nueva línea. Aquí está una lista parcial.

| | |
|---|---|
| \a | Alerta audible (campanilla) |
| \b | Espacio hacia atrás (backspace) |
| \f | Salto de página |
| \n | Carácter de nueva línea (avance de línea) |
| \t | Carácter tabulador |
| \r | Carácter de retorno de carro |
| \v | Tabulador vertical |

# 3 Especificaciones de Formato

El verdadero potencial de printf se muestra cuando imprimimos el contenido de variables. Tomemos el especificador de formato %d por ejemplo. Esto imprime un número. Por lo tanto, se debe proporcionar un número para imprimir. Esto se hace agregando otro argumento a la instrucción printf, como se muestra aquí.

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

En este ejemplo, printf tiene dos argumentos. El primero es una cadena: "Tengo %d años \n". El segundo es un número entero, la edad.

## 3.1 La lista de argumentos

Cuando printf procesa sus argumentos, comienza a imprimir los caracteres que encuentra en el primer argumento, uno por uno. Cuando encuentra un porcentaje, sabe que tiene una especificación de formato. Va al siguiente argumento y usa su valor, imprimiendo de acuerdo con esa especificación de formato. Luego vuelve a imprimir un carácter a la vez (desde el primer argumento). Está bien incluir más de una especificación de formato en la cadena printf. En ese caso, la primera especificación de formato va con el primer argumento adicional, el segundo va con el segundo, y así sucesivamente. Pongo un ejemplo a continuación:

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

## 3.2 Porcentaje

Cada especificación de formato comienza con un signo de porcentaje y termina con una letra. Las letras se eligen para tener algún significado mnemónico. Aquí está una lista parcial:

| | |
|---|---|
| %c | Imprime un carácter simple |
| %d | Imprime un número decimal (base 10) |
| %e | Imprime un número exponencial de coma flotante |
| %f | Imprime un número de coma flotante |
| %g | Imprime un número de coma flotante en formato general |
| %i | Imprime un entero en base 10 |
| %o | Imprime un número octal (base 8) |
| %s | Imprime una cadena de caracteres |
| %u | Imprime un número decimal sin signo (base 10) |
| %x | Imprime un número hexadecimal (base 16) |
| %% | Imprime el símbolo de % (también funciona \%) |

Para imprimir un número de manera simple, el especificador de formato es simplemente %d. Aquí hay algunos ejemplos de casos y los resultados. Producidos por Printf.

| | |
|---|---|
| ("%d",0) | 0 |
| ("%d",-7) | -7 |
| ("%d",1560133635) | 1560133635 |
| ("%d",-2035065302) | -2035065302 |

Observe que de la manera simple, %d, no hay un tamaño predeterminado para el resultado. Printf simplemente ocupa todo el espacio que necesita.

## 3.3 La opción de anchura

Como mencioné anteriormente, simplemente imprimir números no era suficiente. Había opciones especiales que se deseaban. La más importante fue probablemente la opción de ancho. Al decir %5d, se garantizaba que el número ocuparía cinco espacios (más si fuera necesario, nunca menos). Esto fue muy útil en

la impresión de tablas porque los números pequeños y grandes ocupaban la misma cantidad de espacio. Casi toda la impresión era mono espaciada en esos días, lo que significa que cualquier dato ocupaba la misma cantidad de espacio. Esto todavía es común en los editores de texto utilizados por los programadores. Para imprimir un número con un cierto ancho (mínimo), digamos 5 espacios de ancho, el especificador de formato es %5d. Aquí están algunos ejemplos y resultados (usamos el símbolo para especificar explícitamente un espacio.) producidos por printf.

| Printf | Produce |
|---|---|
| ("%d",0) | 0 |
| ("%d",-7) | -7 |
| ("%d",1560133635) | 1560133635 |
| ("%d",-2035065302) | -2035065302 |

Tenga en cuenta que para números más cortos, el resultado se rellena con espacios iniciales. Para números excesivamente largos no hay relleno, y se imprime el número completo. En uso normal, uno haría el campo lo suficientemente ancho para el número más grande que uno esperaría. Si sus números son generalmente de uno, dos o tres dígitos, entonces %3d probablemente sea adecuado. En un uso erroneo, uno podría terminar imprimiendo un número que es demasiado grande para el campo. Printf toma la decisión de imprimir dichos números por completo, a pesar de que ocupan demasiado espacio. Esto se debe a que es mejor imprimir la respuesta correcta y verse fea que imprimir la respuesta incorrecta y verse bonita

## 3.4 Rellenando el espacio extra

Al imprimir un número pequeño como 27 en un campo% 5d, la pregunta se convirtió en dónde colocar el 27 y qué poner en los otros tres espacios. Podría imprimirse en los primeros dos espacios, los dos últimos espacios, o tal vez los dos espacios del medio (si eso se puede determinar). Los espacios vacíos pueden llenarse con el carácter en blanco, o quizás estrellas (*** 27 o 27 *** o ** 27 *), o signos de dólar ($$$ 27), o signos de igual (=== 27), o iniciales ceros (como 00027). Estos caracteres adicionales a menudo se denominan caracteres de "protección de cheques" porque están destinados a evitar que los malos cambien la cantidad en dólares de un cheque impreso. Es relativamente fácil cambiar un espacio en otra cosa. Es más difícil cambiar una estrella, un signo de dólar o un signo igual. Printf proporciona relleno de espacio (izquierda o derecha) y relleno de cero (solo izquierda). Si desea verificar la protección o el centrado, debe hacer otros arreglos. Pero incluso sin protección de cheques o centrado printf todavía tiene una impresionante (y desconcertante) colección de opciones

## 3.5 La opcion de justificación

El uso de números printf puede justificarse a la izquierda (impreso en el lado izquierdo del campo) o justificarse a la derecha (impreso en el lado derecho del campo). La forma más natural de imprimir números parece estar justificada correctamente con espacios iniciales. Eso es lo que significa %5d: imprima un número de base 10 en un campo de ancho 5, con el número alineado a la derecha y lleno de espacios al frente. Para hacer que el número esté alineado a la izquierda, se agrega un signo menos al especificador de formato. Para imprimir un número de 5

espacios de ancho y justificado a la izquierda (alineado a la izquierda), el especificador de formato es%-5d. Aquí hay algunos ejemplos de casos y resultados.

| Printf | Produce |
|---|---|
| ("%-5d",0) | 0 |
| ("%-5d",-7) | -7 |
| ("%-5d",1560133635) | 1560133635 |
| ("%-5d",-2035065302) | -2035065302 |

Como antes, para números más cortos, el resultado se rellena con espacios. Para números más largos no hay relleno, y el número no se acorta.

## 3.6 La opcion de rellenar con ceros

Para que las cosas se alineen bonitas y bonitas, es común escribir una fecha usando ceros a la izquierda. Podemos escribir el 5 de mayo de 2003 en los Estados Unidos como 05/05/2003. También podríamos escribirlo como 2003.05.05. Tenga en cuenta que en ambos casos, los ceros a la izquierda no cambian el significado. Simplemente hacen que se alineen muy bien en las listas. Cuando un número se llena con cero, los ceros siempre van al frente, y el número resultante se justifica tanto a la izquierda como a la derecha. En este caso, el signo menos no tiene efecto. Para imprimir un número lleno de cero con 5 espacios de ancho, el especificador de formato es% 05d. Aquí hay algunos ejemplos de casos y resultados

| Printf | Produce |
|---|---|
| ("%05d",0) | 00000 |
| ("%05d",-7) | -0007 |
| ("%05d",1560133635) | 1560133635 |
| ("%05d",-2035065302) | -2035065302 |

Los números más cortos se rellenan con ceros a la izquierda. Los números más largos no cambian.

## 3.7 Diversión con signos más

Los números negativos siempre se imprimen con un signo menos. Los números positivos y cero generalmente no se imprimen con un signo, pero puede solicitar uno. Un signo más (+) en el especificador de formato hace esa solicitud. Para imprimir un número con signo de 5 espacios de ancho, el especificador de formato es %+5d. Aquí hay algunos ejemplos de casos y resultados

| Printf | Produce |
|---|---|
| ("%+5d",0) | +0 |
| ("%+5d",-7) | -7 |
| ("%+5d",1560133635) | +1560133635 |
| ("%+5d",-2035065302) | -2035065302 |

Observe que el carácter cero se trata como un número positivo. Los números más cortos están rellenados. Los números más largos no cambian. Más y menos no están relacionados. Ambos pueden aparecer en un especificador de formato.

## 3.8 El signo positivo invisible

Este es un poco extraño. Es un signo más invisible. En lugar de imprimir un signo más en números positivos (y cero), imprimimos un espacio donde iría el signo. Esto puede ser útil para imprimir números justificados a la izquierda donde desea que los signos menos realmente se destaquen. Observe estas dos alternativas

| Printf | Produce |
|---|---|
| ("%+-5d",0) | +0 |
| ("%+-5d",-7) | -7 |
| ("%+-5d",1560133635) | +1560133635 |
| ("%+-5d",-2035065302) | -2035065302 |
| ("% -5d",0) | 0 |
| ("% -5d",-7) | -7 |
| ("% -5d",1560133635) | 1560133635 |
| ("% -5d",-2035065302) | -2035065302 |

Recuerde desde arriba que el especificador de formato %-5d obtenemos los siguientes resultados (mostrados nuevamente para una comparación anterior).

| Printf | Produce |
|---|---|
| ("%-5d",0) | 0 |
| ("%-5d",-7) | -7 |
| ("%-5d",1560133635) | 1560133635 |
| ("%-5d",-2035065302) | -2035065302 |

Observe que los signos más desaparecen, pero el signo todavía ocupa espacio en la parte delantera del número. Podemos ver tambien que se pueden combinar varias opciones en el mismo especificador de formato. En este caso, hemos combinado las opciones más, menos y cinco, o espacio, menos y cinco, o simplemente menos y cinco.

## 3.9 Signo más, espacio y cero

Podemos ver aquí otro ejemplo de combinar varias opciones al mismo tiempo. Usando el especificador de formato %05d obtenemos los siguientes resultados

| Printf | Produce |
|---|---|
| ("%05d",0) | 0000 |
| ("%05d",-7) | -0007 |
| ("%05d",1560133635) | 1560133635 |

| | |
|---|---|
| ("%05d",-2035065302) | -2035065302 |

Usando los especificadores de formato %+05d o %0+5d podremos obtener los siguientes resultados.

| Printf | Produce |
|---|---|
| ("%+05d",0) | +0000 |
| ("%+05d",-7) | -0007 |
| ("%+05d",1560133635) | +1560133635 |
| ("%+05d",-2035065302) | -2035065302 |

Cuando combinamos el signo más y espacio al mismo tiempo, el espacio organiza espacio para un signo y el más lo usa. Es lo mismo que si el espacio ni siquiera se especificara. El signo más tiene prioridad sobre el espacio.

## 3.10 En resumen

Las opciones también son denominadas "banderas" o “flags” pueden aparecer y combinarse en cualquier orden. A continuación listamos algunas de ellas.

| Bandera o flag | Efecto en la impresión |
|---|---|
| Ninguno | Imprime normal (alineado a la derecha relleno de espacios) |
| - | Justificación a la izquierda |
| 0 | Rellenos con ceros |
| + | Imprime un más en los números positivos |
| Spc | Imprime un signo más invisible |

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

# 4 Imprimiendo cadenas

La opción %s nos permite imprimir una cadena dentro de una cadena. Aquí hay un ejemplo

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

El indicador de justificación a la izquierda se aplica a las cadenas, pero, por supuesto, el relleno cero, el signo más y el signo más invisible significan menos

| Printf | Produce |
|---|---|
| ("%5s","") | |
| ("%5s","a") | a |

| | |
|---|---|
| ("%5s","ab") | ab |
| ("%5s","abcdefg") | abcdefg |
| ("%-5s","") | |
| ("%-5s","a") | a |
| ("%-5s","ab") | ab |
| ("%-5s","abcdefg") | abcdefg |

# 5 Coma flotante

Los números de coma flotante son aquellos como 3.1415 que tienen un punto decimal en algún lugar dentro. Esto está en contraste con los enteros ordinarios como 27 que no tienen punto decimal. Se aplican las mismas “banderas o flags” y reglas para números de punto flotante que para números enteros, pero tenemos algunas opciones nuevas. Lo más importante es una forma de especificar cuántos dígitos aparecen después del punto decimal. Este número se llama precisión del número. Para el comercio ordinario, los precios a menudo se mencionan como dólares enteros o como dólares y centavos (cero o dos dígitos de precisión). Para la gasolina, los precios se mencionan en dólares, centavos y décimas de centavo (tres dígitos de precisión). Aquí hay algunos ejemplos de cómo imprimir este tipo de números. Para la asignación e=2.718281828.

| Printf | Produce |
|---|---|
| ("%.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 |

Tenga en cuenta que si se especifican un punto y un número, el número (la precisión) indica cuántos lugares se deben mostrar después del punto decimal. Observe que si no se especifican puntos y precisión para %f, el valor predeterminado es % .6f (seis dígitos después del punto decimal). Observe que si se especifica una precisión de cero, el punto decimal también desaparece. Si desea recuperarlo, debe enumerarlo por separado (después del especificador de formato %f). Podemos especificar tanto un ancho como una precisión al mismo tiempo. Observe especialmente que 5.2 significa un ancho total de cinco, con dos dígitos después del decimal. Es muy común y natural pensar que significa cinco dígitos delante del decimal y dos dígitos después, pero eso no es correcto. Tengan cuidado.

| Printf | Produce |
|---|---|
| ("%5.0f",e) | 3 |
| ("%5.0f.",e) | 3. |
| ("%5.1f",e) | 2.7 |

| | |
|---|---|
| ("%5.2f",e) | 2.72 |
| ("%5.7f",e) | 2.7182818 |

También podemos combinar la precisión con los indicadores que aprendimos antes, para especificar la justificación izquierda, los ceros a la izquierda, los signos más, etc

| Printf | Produce |
|---|---|
| ("%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 |

# 6 Diseñando la especificación perfecta

Si está diseñando un especificador de formato printf, el primer paso es decidir qué tipo de cosa está imprimiendo. Si es un número entero, un flotador, una cadena o un carácter, tomará diferentes decisiones sobre qué formato básico usar. La segunda gran pregunta es qué tan ancho debe ser su campo. Por lo general, este será el tamaño del número más grande que espera imprimir en circunstancias normales. A veces, esto se controla por la cantidad de espacio que se proporciona en un formulario preimpreso (como un cheque o una factura). Decida qué desea imprimir en una variedad de circunstancias. En este documento, a menudo hemos ilustrado los resultados utilizando un número positivo pequeño, un número negativo pequeño, un número positivo de gran tamaño y un número negativo de gran tamaño. Debe incluir estas opciones, así como números grandes (pero no demasiado grandes). Diseñe su formato para el número más grande que normalmente esperaría ver.

# 7 Recomendaciones para la prueba

La prueba del printf incluye una variedad de problemas de coincidencia. Están diseñados para ser difíciles, y los comentarios de los estudiantes indican que, en todo caso, son más difíciles de lo esperado. Puede utilizar el proceso de eliminación para hacer que esta prueba sea muy rápida y precisa. Al observar una característica común en la línea de respuesta, puede descartar todas esas declaraciones printf que no tienen esa característica. Muy rápidamente puede reducir sus opciones a una o dos.

## 7.1 Características sencillas

Es fácil comprobar si los números cortos tienen ceros a la izquierda. Si es así, debe haber un cero en la especificación de formato. Es fácil ver si los números positivos tienen signos más. Si es así, debe haber un signo positivo en la especificación de formato.

## 7.2 Antes, Entre y Tras

Lo siguiente a tener en cuenta es el contenido que tenemos antes, entre y tras el número que se imprime. En una especificación de formato como x%5dz hay una x antes del número y una z detrás del número. Las x y z no son parte de la especificación de formato, pero son parte del resultado impreso. Todo lo demás que se imprime se encuentra "entre". Para decidir si hay algo antes o detrás de un número, observe el número negativo sobre dimensionado. Cualquier espacio que aparezca antes lo hará seguramente antes de la especificación de formato. Cualquier espacio detrás de él seguramente está detrás de la especificación de formato. A continuación tenemos un ejemplo. Si -2035065302 se imprime tal que “~~bb~~-2035065302” puedes tener la seguridad de que la cadena %... ,se encuentra con dos espacios antes de la especificación de formato y un espacio tras ella. Esto se debe a que todas las posiciones de impresión entre (el % y lo que corresponda) son utilizadas por el número de gran tamaño. Una vez que haya determinado lo que está antes y detrás, puede usar esa información para compararla con las opciones de coincidencia. A menudo, esto le dará la respuesta directamente

## 7.3 El signo invisible de suma

Compare el número negativo sobre dimensionado con el número positivo sobre dimensionado. Si el número positivo tiene un espacio adicional al frente, es un signo más invisible. Si no hay espacio adicional, no hay un signo más invisible.

## 7.4 Justificación izquierda

Elimina lo que hay antes y lo de después, observa lo que queda. Mira el pequeño número negativo, ¿donde se imprimen los espacios adicionales? Si se encuentran delante del numero, el número estará justificado correctamente. Si están detrás, el número queda justificado. Si se encuentran en ambos extremos, hiciste algo mal.

# 8 Conclusión

La función printf es un poderoso dispositivo para imprimir números y otros elementos almacenados en variables. Todo esto implica cierta complejidad. Tomado de una vez, la complejidad hace que printf parezca casi imposible de entender. Pero la complejidad se puede desplegar fácilmente en características simples, que incluyen ancho, precisión, señalización, justificación y relleno. Al reconocer y comprender estas características, printf se convertirá en un servidor útil y amigable en tus intentos de impresión.