Ve al primero, anterior, próximo, último sección, tabla de contenidos.

Funciones aritméticas

Este capítulo contiene información sobre las funciones para realizar operaciones aritméticas básicas, como dividir un número flotante en sus partes entera y fraccionaria, o recuperar la parte imaginaria de un valor complejo. Estas funciones se declaran en los archivos de encabezado `math.h' y `complex.h'.

números enteros

El lenguaje C define varios tipos de datos enteros: entero, entero corto, entero largo y carácter, todos con y sin signo. El compilador GNU C extiende el lenguaje para que también admita enteros largos.

Los tipos enteros de C se diseñaron para permitir la portabilidad del código entre máquinas con diferentes tamaños de datos inherentes (tamaños de palabra), por lo que cada tipo puede tener rangos diferentes en distintas máquinas. El problema radica en que, a menudo, un programa debe escribirse para un rango específico de enteros y, en ocasiones, para un tamaño de almacenamiento específico, independientemente de la máquina en la que se ejecute.

Para solucionar este problema, la biblioteca GNU C contiene definiciones de tipos de C que permite declarar enteros que se ajusten a sus necesidades específicas. Dado que los archivos de encabezado de la biblioteca GNU C se personalizan para cada máquina específica, el código fuente de su programa no tiene por qué serlo.

Estas definiciones de tipos están en 'stdint.h'.

Si necesita que un número entero se represente en exactamente N bits, utilice uno de los siguientes tipos, con la asignación obvia al tamaño de bit y al signo:

- int8_t
- int16_t
- int32_t
- int64_t
- uint8_t
- uint16_t
- uint32_t
- uint64_t

Si su compilador de C y su máquina de destino no permiten números enteros de un tamaño determinado, el tipo correspondiente mencionado anteriormente no existe.

Si no necesita un tamaño de almacenamiento específico, pero desea la estructura de datos más pequeña con en al menos N bits, usa uno de estos:

- int_menos8_t
- int_menos16_t
- int_menos32_t
- int_menos64_t
- uint_menos8_t

- uint_least16_t
- uint_least32_t
- uint_least64_t

Si no necesita un tamaño de almacenamiento específico, pero desea la estructura de datos que lo permita, el acceso más rápido mientras tenga al menos N bits (y entre estructuras de datos con la misma velocidad de acceso, la más pequeña), utilice una de estas:

- int_fast8_t
- int_fast16_t
- int_fast32_t
- int_fast64_t
- uint_fast8_t
- uint_fast16_t
- uint_fast32_t
- uint_fast64_t

Si desea un número entero con el rango más amplio posible en la plataforma en la que se encuentra Si se utiliza, utilice uno de los siguientes. Si los utiliza, debe escribir código que tiene en cuenta el tamaño variable y el rango del entero.

- intmax_t
- uintmax_t

La biblioteca GNU C también proporciona macros que le indican el valor máximo y mínimo Valores posibles para cada tipo de dato entero. Los nombres de las macros siguen estos ejemplos: INT32_MAX, UINT8_MAX, INT_FAST32_MIN, INT_LEAST64_MIN, UINTMAX_MAX, INTMAX_MAX, INTMAX_MIN. Tenga en cuenta que no existen macros para los mínimos de enteros sin signo. Estos son Siempre cero.

Hay macros similares para usar con los tipos enteros integrados de C que deberían venir con su compilador de C. Estos se describen en la sección Mediciones de tipos de datos.

No olvides que puedes usar la función sizeof de C con cualquiera de estos tipos de datos para obtener la cantidad de bytes de almacenamiento que utiliza cada uno.

División de enteros

Esta sección describe funciones para realizar divisiones enteras. Estas funciones son redundantes cuando se utiliza GNU CC, porque en GNU C el operador `/' siempre Se redondea hacia cero. Pero en otras implementaciones de C, "/" puede redondearse de forma diferente. con argumentos negativos. div y ldiv son útiles porque especifican cómo redondear El cociente: hacia cero. El resto tiene el mismo signo que el numerador.

Estas funciones se especifican para devolver un resultado .rem o tal que el valor o .quot* denominador + o igual a numerador .

Para utilizar estas funciones, debe incluir el archivo de encabezado 'stdlib.h' en su programa.

Tipo de dato: div_t

Este es un tipo de estructura que se utiliza para almacenar el resultado devuelto por la función div . tiene los siguientes miembros:

int comillas

El cociente de la división.

entero rem
El resto de la división.

Función: div_t div Esta (denominador numerador int) int
función div calcula el cociente y el resto de la división de
numerador por denominador , devolviendo el resultado en una estructura de tipo div_t.

Si el resultado no se puede representar (como en una división por cero), el comportamiento es indefinido.

He aquí un ejemplo, aunque no muy útil.

```
div_t resultado;
resultado = div (20, -6);
```

Ahora result.quot es -3 y result.rem es 2.

Tipo de dato: ldiv_t
Este es un tipo de estructura que se utiliza para almacenar el resultado devuelto por la función ldiv . tiene los siguientes miembros:

int largo comillas
El cociente de la división.

largo int rem
El resto de la división.

(Esto es idéntico a div_t excepto que los componentes son de tipo long int en lugar de que int.)

Función: ldiv_t ldiv La (denominador largo largo int numerador)
función ldiv es similar a div, excepto que los argumentos son de tipo long
int y el resultado se devuelve como una estructura de tipo ldiv_t.

Tipo de dato: lldiv_t
Este es un tipo de estructura utilizado para almacenar el resultado devuelto por la función lldiv .
Está compuesto por los siguientes miembros:

largo largo int quot
El cociente de la división.

largo largo int rem
El resto de la división.

(Esto es idéntico a div_t excepto que los componentes son de tipo long long int en lugar de int.)

Función: lldiv_t lldiv La (denominador largo largo int numerador largo largo)
función lldiv es como la función div , pero los argumentos son de tipo long
int largo y el resultado se devuelve como una estructura de tipo lldiv_t.

La función lldiv se agregó en ISO C99.

Tipo de dato: imaxdiv_t
Este es un tipo de estructura utilizado para almacenar el resultado devuelto por la función imaxdiv .
Está compuesto por los siguientes miembros:

intmax_t quot
El cociente de la división. intmax_t rem El resto de la
división.

(Esto es idéntico a div_t excepto que los componentes son de tipo intmax_t en lugar de int).

Ver sección Números enteros para una descripción del tipo intmax_t .

Función: imaxdiv_t imaxdiv La (intmax_t numerador, intmax_t denominador)
función imaxdiv es como la función div , pero los argumentos son de tipo intmax_t y el resultado se devuelve como una estructura de tipo imaxdiv_t.

Ver sección Números enteros para una descripción del tipo intmax_t .

La función imaxdiv se agregó en ISO C99.

Números de punto flotante

La mayoría del hardware informático admite dos tipos de números: enteros (@math{...-3, -2, -1, 0, 1, 2, 3...}) y números de punto flotante. Los números de punto flotante constan de tres partes: la mantisa, el exponente y el bit de signo. El número real representado por un valor de punto flotante se da mediante @ifnottex @math{(s ? -1 : 1) @mul{} 2^e @mul{} M}, donde @math{s} es el bit de signo, @math{e} el exponente y @math{M} la mantisa. Consulte la sección " Conceptos de representación en punto flotante". para más detalles. (Es posible tener una base diferente para el exponente, pero todo el hardware moderno usa @math{2}.)

Los números de punto flotante pueden representar un subconjunto finito de los números reales. Si bien este subconjunto es suficientemente grande para la mayoría de los propósitos, es importante recordar que los únicos números reales que pueden representarse con exactitud son los números racionales cuya expansión binaria terminal es menor que la anchura de la mantisa. Incluso fracciones simples como @math{1/5} solo pueden aproximarse mediante el punto flotante.

Las operaciones y funciones matemáticas a menudo requieren valores no representables. A menudo, estos valores pueden aproximarse con suficiente precisión para fines prácticos, pero a veces no. Históricamente, no había forma de determinar cuándo los resultados de un cálculo eran inexactos. Las computadoras modernas implementan el estándar IEEE 754 para cálculos numéricos, que define un marco para indicar al programa cuándo los resultados del cálculo no son fiables. Este marco consiste en un conjunto de excepciones que indican por qué un resultado no se pudo representar, y los valores especiales infinito y no un número (NaN).

Funciones de clasificación de números de punto flotante

La norma ISO C99 define macros que permiten determinar qué tipo de número de punto flotante contiene una variable.

Macro: int fpclassify. Esta (tipo flotante x)
macro genérica funciona con todos los tipos de coma flotante y devuelve un valor de tipo int. Los valores posibles son:

FP_NAN
El número de punto flotante NaN) incógnita "No es un número" (ver sección Infinito y

FP_INFINITO
El valor de NaN) incógnita es más o menos infinito (ver sección Infinito y

FP_ZERO
El valor de es incógnita cero. En formatos de punto flotante como IEEE 754, donde cero se puede firmar, este valor también se devuelve si es cero negativo.

FP_SUBNORMAL
Números cuyo valor absoluto es demasiado pequeño para ser representado en la El formato normal se representa en un formato alternativo desnormalizado. (ver sección Conceptos de representación de punto flotante). Este formato es menos preciso pero puede representar valores más cercanos a cero. fpclassify devuelve esto valor para valores de incógnita en este formato alternativo.

FP_NORMAL
Este valor se devuelve para todos los demás valores sin nada incógnita. Indica que hay especial acerca del número.

fpclassify es más útil si se debe probar más de una propiedad de un número.
Existen macros más específicas que solo prueban una propiedad a la vez. Generalmente Estas macros se ejecutan más rápido que fpclassify, ya que hay soporte de hardware especial Para ellos. Por lo tanto, deberías usar las macros específicas siempre que sea posible.

Macro: int isfinite. Esta (tipo flotante x)
macro devuelve un valor distinto de cero si no es incógnita es finito: no más ni menos infinito, y NaN. Es equivalente a

```
(fpclassify(x) != FP_NAN && fpclassify(x) != FP_INFINITE)
```

isfinite se implementa como una macro que acepta cualquier tipo de punto flotante.

Macro: int es normal (tipo flotante x)
Esta macro devuelve un valor distinto de cero si es incógnita es finito y normalizado. Es equivalente a

```
(fpclassify(x) == FP_NORMAL)
```

Macro: int isnan (tipo flotante x)
Esta macro devuelve un valor distinto de cero si incógnita es NaN. Es equivalente a

```
(fpclassify(x) == FP_NAN)
```

BSD proporcionó otro conjunto de funciones de clasificación de punto flotante.
La biblioteca GNU C también admite estas funciones; sin embargo, recomendamos que las utilice Las macros ISO C99 en el nuevo código. Son estándar y estarán disponibles más adelante.
ampliamente. Además, como son macros, no tienes que preocuparte por el tipo de su argumento.

Función: int isinf Función: (doble x)
int isinff Función: int isinfl (flotantex)
Esta función devuelve -1 (doble x larga)
si es infinito positivo y 0 en caso incógnita representa el infinito negativo, 1 si incógnita representa contrario.

Función: int isnan Función: (doble x)
int isnanf Función: int isnanl (flotantex)
Esta función devuelve un (doble x larga)
valor distinto de cero en caso contrario. Incógnita es un valor "no numérico" y cero

Nota: La macro isnan definida por ISO C99 anula la función BSD. Esto es
Normalmente no es un problema, porque las dos rutinas se comportan de forma idéntica. Sin embargo,
Si realmente necesitas obtener la función BSD por alguna razón, puedes escribir

(esnan) (x)

Función: int finite Función: (doble x)
int finitef Función: int finitel (flotantex)
Esta función devuelve un (doble x larga)
valor distinto de cero si y cero en caso contrario. Incógnita es finito o un valor "no numérico",

Función: double infnan. Esta (errorint)
función se proporciona por compatibilidad con BSD. Su argumento es un error.
código, EDOM o ERANGE; infnan devuelve el valor que devolvería una función matemática
Si se establece errno en ese valor. Consulte la sección " Información de errores por parte de Mathematical".
Funciones. -ERANGE también es aceptable como argumento y corresponde a
-HUGE_VAL como valor.

En la biblioteca BSD, en ciertas máquinas, infnan genera una señal fatal en todos los casos.
La biblioteca GNU no hace lo mismo, porque no se ajusta a la norma ISO C.
especificación.

Nota de portabilidad: Las funciones enumeradas en esta sección son extensiones BSD.

Errores en los cálculos de punto flotante

Excepciones de FP

El estándar IEEE 754 define cinco excepciones que pueden ocurrir durante una
cálculo. Cada uno corresponde a un tipo particular de error, como el desbordamiento.

Cuando ocurren excepciones (cuando se plantean excepciones, en el lenguaje del
estándar), pueden ocurrir dos cosas. Por defecto, la excepción simplemente se anota.
en la palabra de estado de punto flotante, y el programa continúa como si nada hubiera sucedido.
sucedió. La operación produce un valor predeterminado, que depende de la excepción.
(ver la tabla a continuación). Su programa puede verificar la palabra de estado para averiguar qué
Se produjeron excepciones.

Alternativamente, puede habilitar trampas para excepciones. En ese caso, cuando una excepción...
Si se activa, el programa recibirá la señal SIGFPE . La acción predeterminada para esto
La señal es para terminar el programa. Consulte la sección Manejo de señales para saber cómo...
cambiar el efecto de la señal.

En la biblioteca matemática System V, la función definida por el usuario matherr se llama cuando
Se producen ciertas excepciones dentro de las funciones de la biblioteca matemática. Sin embargo, Unix98
El estándar desaprueba esta interfaz. La admitimos por compatibilidad histórica, pero
Te recomiendo que no lo uses en programas nuevos.

Las excepciones definidas en IEEE 754 son:

'Operación no válida'
Esta excepción se genera si los operandos dados no son válidos para la operación que se va a realizar. Algunos ejemplos son (véase IEEE 754, sección 7):

1. Suma o resta: @math{@infinity{} - @infinity{}}. (Pero @math{@infinito{} + @infinito{} = @infinito{}}).
2. Multiplicación: @math{0 @mul{} @infinity{}}.
3. División: @math{0/0} o @math{@infinity{}/@infinity{}}.
4. Resto: @math{x} REM @math{y}, donde @math{y} es cero o @math{x} es infinito.
5. Raíz cuadrada si el operando es menor que cero. En general, cualquier función matemática evaluada fuera de su dominio produce esta excepción.
6. Conversión de un número de punto flotante a una cadena entera o decimal, cuando el número no se puede representar en el formato de destino (debido a desbordamiento, infinito o NaN).
7. Conversión de una cadena de entrada irreconocible.
8. Comparación mediante predicados que involucran @math{<} o @math{>}, cuando uno de los operandos es NaN. Puede evitar esta excepción utilizando las funciones de comparación no ordenadas; consulte la sección Funciones de comparación de punto flotante.

Si la excepción no atrapa, el resultado de la operación es NaN.

División por cero
Esta excepción se produce cuando un número finito distinto de cero se divide por cero. Si no se produce ninguna trampa, el resultado es @math{+@infinity{}} o @math{-@infinity{}}, según los signos de los operandos.

'Desbordamiento'
Esta excepción se genera cuando el resultado no puede representarse como un valor finito en el formato de precisión del destino. Si no se produce ninguna trampa, el resultado depende del signo del resultado intermedio y del modo de redondeo actual (IEEE 754, sección 7.3).

1. Redondear al más cercano lleva todos los desbordamientos a @math{@infinity{}} con el signo del resultado intermedio.
2. Redondear hacia @math{0} lleva todos los desbordamientos al mayor número finito representable con el signo del resultado intermedio.
3. El redondeo hacia @math{-@infinity{}} lleva desbordamientos positivos al número finito representable más grande y desbordamientos negativos a @math{-@infinity{}}.
4. El redondeo hacia @math{@infinity{}} lleva los desbordamientos negativos al número finito representable más negativo y los desbordamientos positivos a @math{@infinity{}}.

Siempre que se genera la excepción de desbordamiento, también se genera la excepción inexacta.

'Desbordamiento'
La excepción de desbordamiento por debajo del valor límite se genera cuando un resultado intermedio es demasiado pequeño para calcularse con precisión, o si el resultado de la operación, redondeado a la precisión de destino, es demasiado pequeño para normalizarse. Si no se instala ninguna trampa para la excepción de desbordamiento por debajo del valor límite, este se señaliza (mediante el indicador de desbordamiento por debajo del valor límite) solo cuando se detectan valores pequeños y pérdida de precisión. Si no se instala ningún controlador de trampas, la operación continúa con un valor pequeño impreciso, o cero si la precisión de destino no puede contener el resultado exacto pequeño.

`Inexacto'

Esta excepción se señala si un resultado redondeado no es exacto (como al calcular la raíz cuadrada de dos) o si un resultado se desborda sin una trampa de desbordamiento.

Infinito y NaN

Los números de punto flotante IEEE 754 pueden representar infinito positivo o negativo, y NaN (no un número). Estos tres valores surgen de cálculos cuyo resultado no está definido o no se puede representar con precisión. También se puede asignar deliberadamente una variable de punto flotante a cualquiera de ellos, lo cual a veces resulta útil. Algunos ejemplos de cálculos que producen infinito o NaN:

@ifnottex

@math{1/0 = @infinito{}} @math{log (0) =
-@infinito{}} @math{sqrt (-1) = NaN}

Cuando un cálculo produce cualquiera de estos valores, también ocurre una excepción; consulte la sección Excepciones de FP.

Las operaciones básicas y las funciones matemáticas aceptan infinito y NaN y producen resultados lógicos. Los infinitos se propagan a través de los cálculos como cabría esperar: por ejemplo, @math{2 + @infinity{} = @infinity{}}, @math{4/@infinity{} = 0}, atan @math{(@infinity{}) = @pi{}/2}. NaN, por otro lado, afecta cualquier cálculo que lo involucre. A menos que el cálculo produzca el mismo resultado independientemente del valor real que reemplace a NaN, el resultado es NaN.

En las operaciones de comparación, el infinito positivo es mayor que todos los valores excepto él mismo y NaN, y el infinito negativo es menor que todos los valores excepto él mismo y NaN. NaN no está ordenado: no es igual, mayor ni menor que ningún valor. == x es falso si el valor de x es NaN. Puede usar esto incluido él mismo para comprobar si un valor es NaN, pero se recomienda usar la función isnan (consulte la sección " Funciones de clasificación de números en coma flotante"). Además, <, >, <= y >= generarán una excepción cuando se apliquen a NaN.

`math.h' define macros que le permiten establecer explícitamente una variable en infinito o NaN.

Macro: float INFINITY. Una
expresión que representa el infinito positivo. Equivale al valor obtenido mediante operaciones matemáticas como 1.0 / 0.0. -INFINITY representa el infinito negativo.

Puede comprobar si un valor de punto flotante es infinito comparándolo con esta macro. Sin embargo, no se recomienda; en su lugar, utilice la macro isfinite . Consulte la sección " Funciones de clasificación de números de punto flotante".

Esta macro se introdujo en el estándar ISO C99.

Macro: float NAN. Una
expresión que representa un valor que no es un número. Esta macro es una extensión GNU, disponible solo en máquinas que admiten el valor "no es un número", es decir, en todas las máquinas que admiten el punto flotante IEEE.

Puede usar `#ifdef NAN' para probar si la máquina admite NaN. (Por supuesto, debe hacer que las extensiones GNU sean visibles, por ejemplo, definiendo

_GNU_SOURCE, y luego debe incluir 'math.h'.)

IEEE 754 también permite otro valor inusual: cero negativo. Este valor es
se produce cuando se divide un número positivo por infinito negativo, o cuando un
El resultado negativo es menor que los límites de representación. El cero negativo se comporta
idénticamente a cero en todos los cálculos, a menos que pruebe explícitamente el bit de signo con
signbit o copysign.

Examinando la palabra de estado de la FPU

ISO C99 define funciones para consultar y manipular la palabra de estado de punto flotante.
Puede utilizar estas funciones para comprobar si hay excepciones no atrapadas cuando sea necesario.
conveniente, en lugar de preocuparse por ellos en medio de un cálculo.

Estas constantes representan las diversas excepciones de IEEE 754. No todas las FPU informan todos los
Las diferentes excepciones. Cada constante se define si y solo si la FPU que está...
La compilación admite esa excepción, por lo que puede probar la compatibilidad con FPU con `#ifdef'.
Se definen en 'fenv.h'.

FE_INEXACT
: La excepción inexacta.

FE_DIVBYZERO
: La excepción de la división por cero.

DESBORDAMIENTO FE
: La excepción de desbordamiento.

DESBORDAMIENTO DE FE
: La excepción de desbordamiento.

FE_INVÁLIDO
: La excepción no válida.

La macro FE_ALL_EXCEPT es el OR bit a bit de todas las macros de excepción que son
apoyado por la implementación del FP.

Estas funciones le permiten borrar indicadores de excepción, probar excepciones y guardar
y restaurar el conjunto de excepciones marcadas.

Función: int feclearexcept (int excepciones)
: Esta función borra todos los indicadores de excepción admitidos indicados por excepciones.

La función devuelve cero en caso de que la operación haya sido exitosa, un valor distinto de cero
valor de lo contrario.

Función: int feraiseexcept (int excepciones)
: Esta función genera las excepciones admitidas indicadas por excepciones. Si mas
orden en que se establecen las excepciones
elevado no está definido excepto que se trate de desbordamiento (FE_OVERFLOW) o desbordamiento insuficiente (FE_UNDERFLOW)
se generan antes de inexactas (FE_INEXACT). Ya sea por desbordamiento o subdesbordamiento,
También se genera una excepción inexacta que depende de la implementación.

La función devuelve cero en caso de que la operación haya sido exitosa, un valor distinto de cero
valor de lo contrario.

Función: int fetestexcept (int excepciones)
: Pruebe si las banderas de excepción indicadas por el parámetro excepto son

actualmente establecido. Si alguno de ellos lo está, se devuelve un valor distinto de cero que especifica Qué excepciones se establecen. De lo contrario, el resultado es cero.

Para entender estas funciones, imagine que la palabra de estado es una variable entera. llamado . estado feclearexcept es entonces equivalente a `status &= ~excepts' y fetestexcept es Equivalente a `(estado y excepciones)'. La implementación real puede ser muy diferente. por supuesto.

Las banderas de excepción solo se borran cuando el programa lo solicita explícitamente, llamando feclearexcept. Si desea comprobar si hay excepciones en un conjunto de cálculos, Primero deberías borrar todas las banderas. Aquí tienes un ejemplo sencillo de cómo usar fetestexcept:

```
{
  doble f;
  int elevado;
  feclearexcept (FE_TODOS_EXCEPTO);
  f = calcular();
  elevado = fetestexcept (FE_OVERFLOW | FE_INVALID);
  si (elevado y FE_OVERFLOW) { /* ... */ }
  si (elevado y FE_INVALID) { /* ... */ }
  /* ... */
}
```

No se pueden establecer bits explícitamente en la palabra de estado. Sin embargo, se puede guardar todo el... Palabra de estado y restaurarla posteriormente. Esto se hace con las siguientes funciones:

Función: int fegetexceptflag (fexcept_t *flagp, int except)

Esta función almacena en la variable apuntada por un valor definido por la implementación que representa la configuración actual de los indicadores de excepción indicados por excepciones. bandera

La función devuelve cero en caso de que la operación haya sido exitosa, un valor distinto de cero valor de lo contrario.

Función: int fesetexceptflag (constante fexcept_t *flagp, int except)

Esta función restaura las banderas para las excepciones indicadas por excepciones los valores almacenados en la variable apuntada por bandera.

La función devuelve cero en caso de que la operación haya sido exitosa, un valor distinto de cero valor de lo contrario.

Tenga en cuenta que el valor almacenado en fexcept_t no se parece en nada a la máscara de bits devuelto por fetestexcept. Es posible que el tipo ni siquiera sea un entero. No intente... modificar una variable fexcept_t .

Informe de errores mediante funciones matemáticas

Muchas de las funciones matemáticas se definen solo sobre un subconjunto del conjunto real o complejo. números. Incluso si están definidos matemáticamente, su resultado puede ser mayor o menores que el rango representable por su tipo de retorno. Estos se conocen como errores de dominio, desbordamientos y subdesbordamientos, respectivamente. Las funciones matemáticas hacen varias cosas cuando ocurre uno de estos errores. En este manual nos referiremos a la respuesta completa como señalización de un error de dominio, desbordamiento o subdesbordamiento.

Cuando una función matemática sufre un error de dominio, genera una excepción no válida y devuelve NaN. También establece error a EDOM; esto es por compatibilidad con sistemas antiguos que

No son compatibles con la gestión de excepciones IEEE 754. Asimismo, cuando se produce un desbordamiento, las funciones matemáticas generan la excepción de desbordamiento y devuelven @math{@infinity{}} o @math{-@infinity{}}, según corresponda. También establecen que si se produce un error a ERANGE. Cuando hay desbordamiento desbordamiento, se genera la excepción de desbordamiento por debajo de la línea y se devuelve cero (con el signo correspondiente). Se puede configurar en ERANGE, pero no está garantizado.

Algunas funciones matemáticas se definen matemáticamente para generar un valor complejo en partes de sus dominios. El ejemplo más común es calcular la raíz cuadrada de un número negativo. Las funciones matemáticas complejas, como csqrt, devolverán el valor complejo apropiado en este caso. Las funciones con valores reales, como sqrt, indicarán un error de dominio.

Algunos hardware antiguos no admiten valores infinitos. En ese hardware, los desbordamientos devuelven un número muy grande (normalmente el mayor representable). `math.h' define macros que permiten comprobar si hay desbordamientos tanto en hardware antiguo como en nuevo.

Macro: double HUGE_VAL
Macro: float HUGE_VALF
Macro: long double HUGE_VALL
Una expresión que representa un número muy grande. En máquinas que usan el formato de punto flotante IEEE 754, HUGE_VAL es infinito. En otras máquinas, suele ser el mayor número positivo que se puede representar.

Las funciones matemáticas devuelven la versión apropiadamente tipificada de HUGE_VAL o -HUGE_VAL cuando el resultado es demasiado grande para ser representado.

Modos de redondeo

Los cálculos de punto flotante se realizan internamente con mayor precisión y luego se redondean para ajustarse al tipo de destino. Esto garantiza que los resultados sean tan precisos como los datos de entrada. IEEE 754 define cuatro posibles modos de redondeo:

Redondear al más cercano.
Este es el modo predeterminado. Debe usarse a menos que se requiera específicamente uno de los otros. En este modo, los resultados se redondean al valor representable más cercano. Si el resultado se encuentra a medio camino entre dos valores representables, se elige el valor representable par. En este caso, "par" significa que el bit de orden más bajo es cero. Este modo de redondeo evita el sesgo estadístico y garantiza la estabilidad numérica: los errores de redondeo en un cálculo largo serán inferiores a la mitad de FLT_EPSILON.

Redondear hacia más infinito.
Todos los resultados se redondean al valor representable más pequeño que sea mayor que el resultado.

Redondear hacia menos infinito.
Todos los resultados se redondean al valor más grande representable que sea menor que el resultado.

Redondear hacia cero.
Todos los resultados se redondean al mayor valor representable cuya magnitud sea menor que la del resultado. En otras palabras, si el resultado es negativo, se redondea hacia arriba; si es positivo, se redondea hacia abajo.

`fenv.h' define constantes que puedes usar para hacer referencia a los distintos redondeos

modos. Cada uno se definirá si y solo si la FPU admite el modo correspondiente.
modo de redondeo.

FE_TONEAREST
Redondear al más cercano.

FE_HACIA ARRIBA
Redondear hacia @math{+@infinity{}}.

FE_HACIA ABAJO
Redondear hacia @math{-@infinity{}}.

FE_HACIA CERO
Redondear hacia cero.

El subdesbordamiento es un caso inusual. Normalmente, los números de punto flotante IEEE 754 son...
siempre normalizado (ver sección Conceptos de representación en punto flotante). Números
menor que @math{2^r} (donde @math{r} es el exponente mínimo,
FLT_MIN_RADIX-1 para flotar) no se puede representar como números normalizados. Redondeo de todos
tales números a cero o @math{2^r} provocarían que algunos algoritmos fallaran en 0.
Por lo tanto, se dejan en forma desnormalizada, lo que produce pérdida de precisión.
ya que se roban algunos bits de la mantisa para indicar el punto decimal.

Si un resultado es demasiado pequeño para ser representado como un número desnormalizado, se redondea a
cero. Sin embargo, el signo del resultado se conserva; si el cálculo fue negativo,
El resultado es cero negativo. El cero negativo también puede resultar de algunas operaciones en
Infinito, como @math{4/-@infinity{}}. El cero negativo se comporta de forma idéntica al cero.
excepto cuando se utilizan las funciones copysign o signbit para comprobar el bit de signo directamente.

En cualquier momento se selecciona uno de los cuatro modos de redondeo anteriores. Puede averiguarlo
¿Cuál con esta función?

Función: int fegetround (vacío)
Devuelve el modo de redondeo seleccionado actualmente, representado por uno de los valores de los macros de modo de redondeo definidos.

Para cambiar el modo de redondeo, utilice esta función:

Función: int fesetround (ronda int)
Cambia el modo de redondeo seleccionado actualmente a redondo. Si redondo no corresponden a uno de los modos de redondeo admitidos, no se cambia nada.
fesetround devuelve cero si cambió el modo de redondeo, un valor distinto de cero si el modo no es compatible.

Si es posible, evite cambiar el modo de redondeo. Puede resultar costoso.
operación; además, algunos hardware requieren que compiles tu programa de manera diferente
Para que funcione. El código resultante podría ser más lento. Consulta tu compilador.
documentación para más detalles.

Funciones de control de punto flotante

Las implementaciones de punto flotante IEEE 754 permiten al programador decidir si
Se producirán trampas para cada una de las excepciones, al establecer bits en la palabra de control. En
C, las trampas hacen que el programa reciba la señal SIGFPE ; consulte la sección Señal
Manejo.

Nota: IEEE 754 dice que a los manejadores de trampas se les proporcionan detalles de las excepciones

situación y puede establecer el valor del resultado. Las señales C no proporcionan ningún mecanismo para pasar esta información de ida y vuelta. Por lo tanto, atrapar excepciones en C no es muy útil.

A veces es necesario guardar el estado de la unidad de punto flotante mientras Realizar algún cálculo. La biblioteca proporciona funciones que guardan y restauran las banderas de excepción, el conjunto de excepciones que generan trampas y el redondeo modo. Esta información se conoce como entorno de punto flotante.

Las funciones para guardar y restaurar el entorno de punto flotante utilizan una variable de tipo fenv_t para almacenar información. Este tipo se define en `fenv.h'. Su tamaño y El contenido está definido por la implementación. No debe intentar manipular un variable de este tipo directamente.

Para guardar el estado de la FPU, utilice una de estas funciones:

Función: int fegetenv (fenv_t *envp)
Almacene el entorno de punto flotante en la variable apuntada por envp .

La función devuelve cero en caso de que la operación haya sido exitosa, un valor distinto de cero valor de lo contrario.

Función: int feholdexcept (fenv_t *envp)
Almacene el entorno de punto flotante actual en el objeto al que apunta. Luego, borre todos envp . los indicadores de excepción y configure la FPU para que no detecte excepciones. No todas Las FPU no admiten la captura de excepciones; si feholdexcept no puede configurar este modo, Devuelve un valor distinto de cero. Si tiene éxito, devuelve cero.

Las funciones que restauran el entorno de punto flotante pueden tomar este tipo de argumentos:

- Punteros a objetos fenv_t , que se inicializaron previamente mediante una llamada a fegetenv o feholdexcept.
- La macro especial FE_DFL_ENV que representa el entorno de punto flotante tal como estaba disponible al inicio del programa.
- La implementación definió macros con nombres que comienzan con FE_ y que tienen tipo fenv_t *. Si es posible, la biblioteca GNU C define una macro FE_NOMASK_ENV que Representa un entorno donde cada excepción generada provoca una trampa. Puede probar esta macro usando #ifdef. Solo se define si _GNU_SOURCE es definido. Algunas plataformas podrían definir otros entornos predefinidos.

Para configurar el entorno de punto flotante, puede utilizar cualquiera de estas funciones:

Función: int fesetenv (constante fenv_t *envp)
Establezca el entorno de punto flotante en el descrito por envp .

La función devuelve cero en caso de que la operación haya sido exitosa, un valor distinto de cero valor de lo contrario.

Función: int feupdateenv (constante fenv_t *envp)
Al igual que fesetenv, esta función establece el entorno de punto flotante en ese descrito por . Sin embargo, si se marcaron excepciones en la palabra de estado envp Antes de llamar a feupdateenv , permanecen marcados después de la llamada. En otros casos Es decir, después de llamar a feupdateenv , la palabra de estado es el OR bit a bit de la Palabra de estado anterior y la guardada en envp .

La función devuelve cero en caso de que la operación haya sido exitosa, un valor distinto de cero valor de lo contrario.

Para controlar excepciones individuales si al generarlas se produce una trampa, puede Utilice las dos funciones siguientes.

Nota de portabilidad: todas estas funciones son extensiones GNU.

Función: int feenableexcept Esta función (int exceptúa)
habilita trampas para cada una de las excepciones según lo indicado por el parámetro excepto Las excepciones individuales se describen en la sección Examinar la palabra de estado de la FPU. Sólo se habilitan las excepciones especificadas, El estado de las demás excepciones no se modifica.

La función devuelve las excepciones habilitadas previamente en caso de que se realice la operación. tuvo éxito, -1 en caso contrario.

Función: int fedisableexcept Esta función (int exceptúa)
deshabilita las trampas para cada una de las excepciones según lo indicado por el parámetro excepto Las excepciones individuales se describen en la sección Examinar la palabra de estado de la FPU. Sólo se deshabilitan las excepciones especificadas. El estado de las demás excepciones no se modifica.

La función devuelve las excepciones habilitadas previamente en caso de que se realice la operación. tuvo éxito, -1 en caso contrario.

Función: int fegetexcept (int exceptúa)
La función devuelve una máscara de bits de todas las excepciones habilitadas actualmente. Devuelve -1 en caso de fallo.

Funciones aritméticas

La biblioteca C proporciona funciones para realizar operaciones básicas con números de punto flotante. Estos incluyen valor absoluto, máximo y mínimo, normalización, manipulación de bits, redondeo y algunos otros.

Valor absoluto

Estas funciones se proporcionan para obtener el valor absoluto (o magnitud) de un número. El valor absoluto de un número real si es positivo, -negativo. Para un número complejo cuya parte real es y cuya parte imaginaria es y , El valor absoluto es sqrt (z, $y^2 + y^2$).

Los prototipos para abs, labs y llabs están en `stdlib.h'; imaxabs se declara en `inttypes.h'; fabs, fabsf y fabsl se declaran en `math.h'. cabs, cabsf y cabsl se declaran en `complejo.h'.

Función: int abs (número entero)
Función: long int labs (números enteros largos)
Función: long long int llabs (número largo) int
Función: intmax_t imaxabs (intmax_t número)
Estas funciones devuelven el valor absoluto de número .

La mayoría de las computadoras utilizan una representación entera en complemento a dos, en la que el

El valor absoluto de INT_MIN (el int más pequeño posible) no se puede representar;
Por lo tanto, abs (INT_MIN) no está definido.

llabs e imaxdiv son nuevos en ISO C99.

Ver sección Números enteros para una descripción del tipo intmax_t .

Función: doble fabulación (número doble)
Función: float fabsf (número flotante)
Función: doble fabsl largo (número doble largo)
Esta función devuelve el valor absoluto del número de punto flotante número .

Función: double cabs Función: (complejo doble z)
float cabsf Función: long (flotante complejo z)
double cabsl Estas funciones devuelven (complejo largo doble z)
el valor absoluto del número complejo (ver z
Sección Números complejos). El valor absoluto de un número complejo es:

raíz cuadrada (creal (z) * creal (z) + cimag (z) * cimag (z))

Esta función debe utilizarse siempre en lugar de la fórmula directa porque
Tiene especial cuidado de no perder precisión. También puede aprovechar
Soporte de hardware para esta operación. Véase la hipótesis en la sección Exponenciación y Logaritmos.

Funciones de normalización

Las funciones descritas en esta sección se proporcionan principalmente como una forma de realizar de manera eficiente ciertas manipulaciones de bajo nivel en números de punto flotante que se representan internamente utilizando un radio binario; consulte la sección Punto flotante Conceptos de representación. Se requiere que estas funciones tengan un comportamiento equivalente Incluso si la representación no utiliza un radio de 2, pero por supuesto es poco probable ser particularmente eficiente en esos casos.

Todas estas funciones se declaran en 'math.h'.

Función: doble frexp (valor doble, int *exponente)
Función: float frexpf (valor flotante, *int exponente)
Función: frexpl doble largo (valor doble largo, int *exponente)
Estas funciones se utilizan para dividir el número y un valor en una fracción normalizada exponente.

Si el argumento multiplicado por una potencia de valor no es cero, el valor de retorno es valor dos, y siempre está en el rango de 1/2 (inclusive) a 1 (excluyendo). El El exponente correspondiente se almacena en exponente ; el valor de retorno multiplicado por 2 * elevado a este exponente es igual al número original valor .

Por ejemplo, frexp (12.8, &exponent) devuelve 0.8 y almacena 4 en exponente.

Si valor es cero, entonces el valor de retorno es cero y cero se almacena en * exponente .

Función: double ldexp (valor doble, exponente int)
Función: float ldexpf (valor flotante, exponente int)
Función: ldexpl doble largo (valor doble largo, exponente int)
Estas funciones devuelven el resultado de multiplicar el número de punto flotante

valor por 2 elevado a la potencia exponente . (Se puede utilizar para volver a ensamblar)
números de punto flotante que fueron desmantelados por frexp.)

Por ejemplo, ldexp (0.8, 4) devuelve 12.8.

Las siguientes funciones, que provienen de BSD, proporcionan facilidades equivalentes a
los de ldexp y frexp.

Función: double logb Función: (doble x)
float logbf Función: long (flotante x)
double logbl Estas funciones devuelven (doble x larga)
la parte entera del logaritmo en base 2 de un entero
Valor representado en tipo doble. Es la potencia entera más alta de 2.
contenido en . Se ignora el signo de . Por ejemplo, logb (3.5) es 1.0 y logb
(4.0) es 2.0.

Cuando 2 elevado a esta potencia se divide en (inclusivo) y 2 , da un cociente entre 1
(exclusivo).

Si es cero, el valor de retorno es menos infinito si la máquina admite infinitos.
y un número muy pequeño si no lo hace. Si es infinito. es infinito, el valor de retorno es

Para finito, el valor devuelto por logb es uno menos que el valor que frexp
se almacenaría en * exponente .

Función: double scalb Función: (valor doble, exponente int)
float scalbf Función: long double (valor flotante, exponente int)
scalbl La función scalb es el nombre BSD (valor doble largo, exponente entero)
para ldexp.

Función: long long int scalbn Función: long (doble n) x, int
long int scalbnf Función: long long int scalbnl (float int n) x,
scalbn es idéntico a scalb, excepto que el (n doble larga) entero x,
exponente es un número de punto flotante. n es un int en lugar de un

Función: long long int scalbln Función: long (doble x, entero largo n)
long int scalblnf Función: long long int scalblnl (flotante x, n larga) int
scalbln es idéntico a scalb, excepto que el (largo doble largo int n)
exponente es un número de punto flotante. n es un int largo en lugar de un

Función: long long int significativo (doble x)
Función: long long int significandf (flotante x)
Función: long long int significativo (doble x larga)
significand devuelve la mantisa del equivalente a scalb escalado al rango @math{[1, 2)}. Es
((double) -ilogb (,)).

Esta función existe principalmente para su uso en ciertas pruebas estandarizadas de IEEE 754.
conformidad.

Funciones de redondeo

Las funciones enumeradas aquí realizan operaciones como redondeo y truncamiento de valores de punto flotante. Algunas de estas funciones convierten números de punto flotante a Valores enteros. Todos están declarados en `math.h'.

También puedes convertir números de punto flotante a enteros simplemente convirtiéndolos en int. Esto descarta la parte fraccionaria, redondeando efectivamente hacia cero. Sin embargo, Esto solo funciona si el resultado puede representarse realmente como un int (para valores muy grandes). números, esto es imposible. Las funciones listadas aquí devuelven el resultado como un doble. En lugar de evitar este problema.

Función: double ceil Función: (doble x)
float ceilf Función: long (flotantex)
double ceill Estas funciones redondean (doble x larga)
hacia arriba al entero más cercano y devuelven ese valor como un doble. Por lo tanto, el techo (1.5) es 2.0.

Función: double floor Función: (doble x)
float floorf Función: long (flotantex)
double floorl Estas funciones redondean (doble x larga)
hacia abajo al entero más cercano y devuelven ese valor. Valor como doble. Por lo tanto, el límite inferior (1,5) es 1,0 y el límite inferior (-1,5) es -2,0.

Función: double trunc Función: (doble x)
float truncf Función: long (flotantex)
double truncl La función trunc redondea (doble x larga)
el formato de punto flotante. Por lo hacia cero al entero más cercano (devuelto en tanto, trunc (1.5) es 1.0 y trunc (-1.5) es -1.0.

Función: double rint Función: (doble x)
float rintf Función: long (flotantex)
double rintl Estas funciones redondean (doble x larga)
el modo. Consulte la sección a un valor entero según el redondeo actual
Parámetros de punto flotante para obtener información sobre...
Varios modos de redondeo. El modo de redondeo predeterminado es redondear al valor más cercano. entero; algunas máquinas admiten otros modos, pero el redondeo al más cercano siempre es posible utilizado a menos que seleccione explícitamente otro.

Si no era inicialmente un entero, estas funciones generan la excepción inexacta.

Función: double nearbyint (doble x)
Función: float nearbyintf (flotantex)
Función: long double nearbyintl (doble x larga)
Estas funciones devuelven el mismo valor que las funciones rint , pero no generan una respuesta. La excepción inexacta si no es un entero.

Función: doble ronda (doble x)
Función: float roundf (flotantex)
Función: redondo doble largo (doble x larga)
Estas funciones son similares a la impresión, pero redondean los casos a la mitad del camino. cero en lugar de al entero par más cercano.

Función: long int lrint (doble x)
Función: long int lrintf (flotantex)
Función: long int lrintl (doble x larga)

Estas funciones son como rint, pero devuelven un int largo en lugar de un número de punto flotante.

Función: long long int llrint Función: long (doble x)
long int llrintf Función: long long int llrintl (flotantex)
Estas funciones son como rint, pero (doble x larga)
devuelven un long long int en lugar de un número de punto flotante.

Función: long int lround (doble x)
Función: long int lroundf (flotantex)
Función: long int lroundl (doble x larga)
Estas funciones son como round, pero devuelven un int largo en lugar de un número de punto flotante.

Función: long long int llround Función: long (doble x)
long int llroundf Función: long long int llroundl (flotantex)
Estas funciones son como round, pero (doble x larga)
devuelven un long long int en lugar de un número de punto flotante.

Función: double modf Función: (doble valor, doble *parte entera)
float modff Función: long (valor flotante, flotante *parte entera)
double modfl Estas funciones dividen el (valor doble largo, doble largo *parte entera)
argumento en una parte entera y una parte fraccionaria. valor
parte (entre -1 y 1, excluyendo). Su suma es igual a . Cada una de las partes valor
tiene el mismo signo que y la parte entera siempre se redondea hacia valor
cero.

modf almacena la parte entera en * ejemplo, modf parte entera , y devuelve la parte fraccionaria. Para (2.5, &intpart) devuelve 0.5 y almacena 2.0 en intpart.

Funciones de resto

Las funciones de esta sección calculan el resto de la división de dos números de punto flotante. Cada una es ligeramente diferente; elija la que mejor se adapte a su problema.

Función: doble fmod (doble numerador, doble denominador)
Función: float fmodf (numerador flotante, denominador flotante)
Función: fmodl doble largo (numerador doble largo, denominador doble largo)
Estas funciones calculan el resto de la división de por numerador
denominador . Específicamente, el valor de retorno es donde es el cociente de dividido por redondeado norte
hacia cero para numerador denominador ,
Un entero. Por lo tanto, fmod (6.5, 2.3) devuelve 1.9, que es 6.5 menos 4.6.

El resultado tiene el mismo signo que la magnitud numerador y tiene una magnitud menor que de la denominador .

Si denominador es cero, fmod señala un error de dominio.

Función: doble sueño (doble numerador, doble denominador)
Función: float dremf (numerador flotante, denominador flotante)
Función: dreml doble largo (numerador doble largo, denominador doble largo)
Estas funciones son como fmod excepto que redondean el cociente interno. norte a

El entero más cercano en lugar de cero a un entero. Por ejemplo, drem (6.5, 2.3) devuelve -0.4, que es 6.5 menos 6.9.

El valor absoluto del resultado es menor o igual a la mitad del valor absoluto de la . La diferencia entre fmod() y drem numerador , denominador (numerador , denominador) siempre es denominador , menos denominador , o cero.

Si denominador es cero, drem señala un error de dominio.

Función: resto doble (doble numerador, doble denominador)
Función: float remainderf (numerador flotante, denominador flotante)
Función: resto doble largo (numerador doble largo, doble largo denominador)

Esta función es otro nombre para drem.

Configuración y modificación de bits individuales de valores FP

Hay algunas operaciones que son demasiado complicadas o costosas de realizar por Manos en números de punto flotante. ISO C99 define funciones para realizar estas operaciones. que en su mayoría implican cambiar bits individuales.

Función: double copysign Función: float (doble doble x, y)
copysignf Función: long double (flotante x, flotador y)
copysignl Estas funciones retornan, pero con el (doble x larga, y doble larga)
signo NaN o cero. Ambas pueden llevar un signo (aunque no y . Trabajan incluso si o y son todas las implementaciones
(apoyarlo) y esta es una de las pocas operaciones que pueden marcar la diferencia.

copysign nunca lanza una excepción.

Esta función está definida en IEC 559 (y el apéndice con las recomendaciones) funciones en IEEE 754/IEEE 854).

Función: int signbit signbit es (tipo flotante x)
una macro genérica que funciona con todos los tipos de punto flotante. Devuelve un valor distinto de cero si el valor de tiene su bit de signo establecido.

Esto no es lo mismo que x < 0,0, porque el punto flotante IEEE 754 permite que el valor de cero sea igual a cero. estar firmado. La comparación -0.0 < 0.0 es falsa, pero signbit (-0.0) devolverá un valor distinto de cero.

Función: doble nextafter (doble doble x, y)
Función: float nextafterf (flotante flotador x, y)
Función: long double nextafterl (doble x larga, y doble larga)

La función nextafter devuelve el siguiente vecino representable de en el dirección hacia . El tamaño y del paso entre y el resultado depende de el tipo del resultado. Si @math{} la función simplemente devuelve Si y . valor es NaN, se devuelve NaN . De lo contrario, un valor correspondiente al valor de El bit menos significativo de la mantisa se suma o se resta, dependiendo de la dirección. nextafter indicará desbordamiento o subdesbordamiento si el resultado va fuera del rango de números normalizados.

Esta función está definida en IEC 559 (y el apéndice con las recomendaciones)

funciones en IEEE 754/IEEE 854).

Función: doble nexttoward (doble x, y doble larga)
Función: float nexttowardf (flotante x, y doble larga)
Función: long double nexttowardl (largo doble largo doble y)
Estas funciones son idénticas a las versiones correspondientes de nextafter excepto que su segundo argumento es un doble largo.

Función: doble nan (constante char *tagp)
Función: float nanf (constante char *tagp)
Función: nanl doble largo (constante char *tagp)
La función nan devuelve una representación de NaN, siempre que NaN sea compatible con la plataforma de destino. nan ("") es equivalente a strtod ("YAYA()"). secuencia de n caracteres

El argumento se utiliza de forma no especificada. En los sistemas IEEE 754, Hay muchas representaciones de NaN y selecciona una. En otras palabras sistemas puede que no haga nada.

Funciones de comparación de punto flotante

Los operadores de comparación estándar de C provocan excepciones cuando uno u otro de los operandos es NaN. Por ejemplo,

```
int v = a < 1.0;
```

generará una excepción si es NaN. (Esto simplemente devuelve no sucede con == y !=; esos falso y verdadero, respectivamente, cuando se examina NaN). Con frecuencia, esto La excepción no es deseable. Por lo tanto, la norma ISO C99 define funciones de comparación que no No se generan excepciones al examinar NaN. Todas las funciones están implementadas. como macros que permiten que sus argumentos sean de cualquier tipo de punto flotante. Las macros Se garantiza que evaluarán sus argumentos solo una vez.

Macro: int isgreater Esta macro (x real flotante, y flotante real)
determina si el argumento es mayor que . Es y equivalente a ( ), pero no se lanza ninguna excepción si son NaN. o y

Macro: int isgreaterequal Esta macro (x real flotante, y flotante real)
determina si el argumento es mayor o igual que Es equivalente a ( ), pero no se genera ninguna excepción si y son NaN.) >= (y o y

Macro: int isless Esta (x real flotante, y flotante real)
macro determina si el argumento es menor que (son NaN). y Es equivalente) < (y), pero no se plantea ninguna excepción si o y

Macro: int es menos igual (x real flotante, y flotante real)
Esta macro determina si el argumento es equivalente a ( ), pero no es menor o igual a son NaN. y . Es se genera ninguna excepción si o y

Macro: int islessgreater Esta macro (x real flotante, y flotante real)
determina si el argumento es menor o mayor que . Es y equivalente a () (aunque solo evalúa y una vez), pero y lanza ninguna excepción si son NaN. o y

Esta macro no es equivalente a != y , porque esa expresión es verdadera si o y

son NaN.

Macro: int isunordered (x real flotante, y flotante real)

determina si sus argumentos están desordenados. En otras palabras,
Es verdadero si x o y son NaN, y falso en caso contrario.

No todas las máquinas ofrecen soporte de hardware para estas operaciones. En las máquinas que...
No lo hagas, las macros pueden ser muy lentas. Por lo tanto, no deberías usar estas funciones.
cuando NaN no es una preocupación.

Nota: No existen macros isequal ni isunequal. Son innecesarias, porque
Los operadores == y != lanzan una excepción si uno o ambos operandos son
no
Yaya.

Funciones aritméticas de FP diversas

Las funciones de esta sección realizan operaciones diversas pero comunes que
son difíciles de expresar con operadores de C. En algunos procesadores, estas funciones pueden
Utilice instrucciones especiales de la máquina para realizar estas operaciones más rápido que la
código C equivalente.

Función: doble fmin (doble doble x, y)
Función: float fminf (flotante flotador x, flotador y)
Función: fminl doble larga (doble x larga, y doble larga)

La función fmin devuelve el menor de los dos valores de la expresión. x y y Es similar a

((x) < (y) ? (x) : (y))

excepto que x y y sólo se evalúan una vez.

Si un argumento es NaN, se devuelve el otro argumento. Si ambos argumentos son
Se devuelve NaN, NaN.

Función: doble fmax (doble doble x, y)
Función: float fmaxf (flotante flotador x, flotador y)
Función: fmaxl doble larga (doble x larga, y doble larga)

La función fmax devuelve el mayor de los dos valores x y y .

Si un argumento es NaN, se devuelve el otro argumento. Si ambos argumentos son
Se devuelve NaN, NaN.

Función: doble fdim (doble doble x, y)
Función: float fdimf (flotante flotador x, flotador y)
Función: fdiml doble largo (doble x larga, y doble larga)

La función fdim devuelve la diferencia positiva entre la diferencia es @math{} si x - y . Lo positivo
es mayor que y , y @math{0} en caso contrario.

Si x, y , o ambos son NaN, se devuelve NaN.

Función: doble fma (doble doble x, y, z flotante)
Función: float fmaf (flotador flotante x, y, z largo)
Función: final doble largo (doble largo doble largo x, y, z doble)

La función fma realiza una multiplicación-suma de punto flotante. Esta es la operación.
@math{(@mul{} }, pero el resultado intermedio no se redondea al

Tipo de destino. Esto a veces puede mejorar la precisión de un cálculo.

Esta función se introdujo porque algunos procesadores tienen una instrucción especial para realizar la multiplicación-suma. El compilador de C no puede usarla directamente, ya que la expresión `x*y + z' está definida para redondear el resultado intermedio. fma permite elegir cuándo se desea redondear solo una vez.

En procesadores que no implementan la multiplicación-suma en hardware, fma puede ser muy lento, ya que debe evitar el redondeo intermedio. `math.h' define los símbolos FP_FAST_FMA, FP_FAST_FMAF y FP_FAST_FMAL cuando la versión correspondiente de fma no es más lenta que la expresión `x*y + z'. En la biblioteca GNU C, esto siempre significa que la operación está implementada en hardware.

Números complejos

La norma ISO C99 introduce compatibilidad con números complejos en C. Esto se logra mediante un nuevo calificador de tipo, complex. Es una palabra clave solo si se incluye `complex.h' .
Hay tres tipos complejos, correspondientes a los tres tipos reales: complejo flotante, complejo doble y complejo doble largo.

Para construir números complejos se necesita una forma de indicar la parte imaginaria de un número. No existe una notación estándar para una constante imaginaria de punto flotante.
En cambio, `complex.h' define dos macros que pueden usarse para crear números complejos.

Macro: const float complex _Complex_I Esta macro es una
representación del número complejo "@math{0+1i}".
Al multiplicar un valor real de punto flotante por _Complex_I se obtiene un número complejo cuyo valor es puramente imaginario. Esto se puede usar para construir constantes complejas:

@math{3.0 + 4.0i} = 3.0 + 4.0 * _Complejo_I

Tenga en cuenta que _Complex_I * _Complex_I tiene el valor -1, pero el tipo de ese valor es complejo.

_Complex_I es un nombre un tanto complicado. 'complex.h' también define un nombre más corto para la misma constante.

Macro: constante flotante compleja I
Esta macro tiene exactamente el mismo valor que _Complex_I. Generalmente es preferible. Sin embargo, causa problemas si se desea usar el identificador I para otra cosa. Se puede escribir con seguridad.

```
#include <complex.h> #undef I
```

Si necesita I para sus propios fines. (En ese caso, le recomendamos que también defina otro nombre corto para _Complex_I, como J).

Proyecciones, conjugados y descomposición de

Números complejos

La norma ISO C99 también define funciones que realizan operaciones básicas con números complejos, como la descomposición y la conjugación. Los prototipos de todas estas funciones son en `complex.h'. Todas las funciones están disponibles en tres variantes, una para cada uno de los tres tipos complejos.

Función: doble crema (complejo doble z)
Función: flotar crealf (flotante complejo z)
Función: pliegue doble largo (complejo largo doble z)
Estas funciones devuelven la parte real del número complejo z.

Función: doble cimag (complejo doble z)
Función: float cimagf (flotante complejo z)
Función: cimagl doble largo (complejo largo doble z)
Estas funciones devuelven la parte imaginaria del número complejo z.

Función: complex double conj Función: (complejo doble z)
complex float conjf Función: complex long (flotante complejo z)
double conjl Estas funciones devuelven el valor (complejo largo doble z)
conjugado del número complejo z.
conjugado de un número complejo tiene la misma parte real y una negada
parte imaginaria. En otras palabras, `conj(a + bi) = a + -bi'.

Función: double carg Función: (complejo doble z)
float cargf Función: long (flotante complejo z)
double cargl Estas funciones devuelven (complejo largo doble z)
el argumento del número complejo. De un número complejo es el ángulo en el plano z. El argumento
complejo entre el positivo y el negativo.
eje real y una recta que pasa por el cero y el número. Este ángulo es
se mide de la manera habitual y varía de @math{0} a
@math{2@pi{}}.

carg tiene una rama cortada a lo largo del eje real positivo.

Función: complex double cproj Función: (complejo doble z)
complex float cprojf Función: complex long (flotante complejo z)
double cprojl Estas funciones devuelven la proyección(complejo largo doble z)
de la esfera de valores complejos. Los valores con una parte imaginaria infinita se z sobre el Riemann
proyectan al infinito positivo.
en el eje real, incluso si la parte real es NaN. Si la parte real es infinita, la
El resultado es equivalente a

```
INFINITO + I * copysign (0.0, cimag (z))
```

Análisis de números

Esta sección describe funciones para "leer" números enteros y de punto flotante.
de una cadena. En algunos casos, puede ser más conveniente usar sscanf o uno de los
funciones relacionadas; consulte la sección Entrada formateada. Pero muchas veces puedes hacer un programa
más robusto al encontrar los tokens en la cadena manualmente y luego convertirlos
números uno por uno.

Análisis de números enteros

Las funciones 'str' se declaran en 'stdlib.h' y las que comienzan con 'wcs' son
Declarado en `wchar.h'. Cabe preguntarse sobre el uso de la restricción en los prototipos.
de las funciones de esta sección. Parece inútil, pero el estándar ISO C utiliza
(para las funciones definidas allí) por lo que tenemos que hacerlo también.

Función: long int strtol (const char *restringir cadena, char **restringir tailptr, int base)

La función strtol ("string-to-long") convierte la parte inicial de un entero con signo, que cadena A un se devuelve como un valor de tipo long int.

Esta función intenta descomponer cadena como sigue:

- Una secuencia (posiblemente vacía) de espacios en blanco. ¿Qué caracteres? Los espacios en blanco están determinados por la función isspace (ver sección Clasificación de personajes). Estos se descartan.
- Un signo más o menos opcional (`+' o `-').
- Una secuencia no vacía de dígitos en el radio especificado por cero, se base . Si base es asume un radio decimal a menos que la serie de dígitos comience con '0' (especificando base octal), o '0x' o '0X' (especificando base hexadecimal); en En otras palabras, la misma sintaxis utilizada para constantes enteras en C. De lo contrario base debe tener un valor entre 2 y 36. Si base es 16, los dígitos pueden Opcionalmente, puede ir precedido de `0x' o `0X'. Si la base no tiene un valor válido, el valor se devuelve 0l y la variable global errno se establece en EINVAL.
- Cualquier carácter restante en la cadena. Si strtol tailptr no es un puntero nulo, almacena un puntero a esta cola en * tailptr .

Si la cadena está vacía, contiene solo espacios en blanco o no contiene una inicial Se realiza una subcadena que tiene la sintaxis esperada para un entero en la conversión base , No especificada. En este caso, strtol devuelve un valor de cero y el El valor almacenado en * es el valor de tailptr cadena .

En una configuración regional distinta a la configuración regional "C" estándar , esta función puede reconocer sintaxis adicional dependiente de la implementación.

Si la cadena tiene una sintaxis válida para un entero pero el valor no es representable Debido al desbordamiento, strtol devuelve LONG_MAX o LONG_MIN (consulte la sección Rango) de un tipo entero), Según corresponda al signo del valor. También establece errno a ERANGE para indicar que hubo desbordamiento.

No debe comprobar si hay errores examinando el valor de retorno de strtol, porque la cadena podría ser una representación válida de 0l, LONG_MAX o LONG_MIN. En su lugar, comprueba si los punteros tailptr apuntan a lo que esperas después del número. (por ejemplo, '\0' si la cadena debe terminar después del número). También debe borrar error antes de la llamada y verificarlo después, en caso de que haya desbordamiento.

Hay un ejemplo al final de esta sección.

Función: int largo wcstol (const wchar_t *restringir cadena, wchar_t **restringir tailptr, entero base)

La función wcstol es equivalente a la función strtol en casi todos los aspectos pero maneja cadenas de caracteres anchos.

La función wcstol se introdujo en la Enmienda 1 de la norma ISO C90.

Función: unsigned long int strtoul (const char *restringir cadena, char **restringir

tailptr, int base)

La función strtoul ("string-to-unsigned-long") es como strtol excepto que convierte a un valor entero largo sin signo . La sintaxis es la misma que la descrita anteriormente para strtol. El valor devuelto en caso de desbordamiento es ULONG_MAX (ver sección Rango de un Tipo entero).

Si cadena representa un número negativo, strtoul actúa igual que strtol pero arroja el resultado es un entero sin signo. Esto significa, por ejemplo, que strtoul está en "-1". devuelve ULONG_MAX y una entrada más negativa que LONG_MIN devuelve (ULONG_MAX + 1) / 2.

conjuntos strtoul error a EINVAL si base está fuera de rango, o ERANGE en desbordamiento.

Función: unsigned long int wcstoul (const wchar_t *restringir cadena, wchar_t **restringir tailptr, base) entero

La función wcstoul es equivalente a la función strtoul en casi todos los aspectos pero maneja cadenas de caracteres anchas.

La función wcstoul se introdujo en la Enmienda 1 de la norma ISO C90.

Función: long long int strtoll (const char *restringir cadena, char **restringir tailptr, int base)

La función strtoll es como strtol excepto que devuelve un valor int largo y acepta números con un rango correspondientemente mayor.

Si la cadena tiene una sintaxis válida para un entero pero el valor no es representable Debido al desbordamiento, strtoll devuelve LONG_LONG_MAX o LONG_LONG_MIN (ver Sección Rango de un Tipo Entero), según corresponda al signo del valor.
También establece errno en ERANGE para indicar que hubo desbordamiento.

La función strtoll se introdujo en ISO C99.

Función: long long int wcstoll (const wchar_t *restringir cadena, wchar_t **restringir punto de cola, entero base)

La función wcstoll es equivalente a la función strtoll en casi todos los aspectos pero maneja cadenas de caracteres anchas.

La función wcstoll se introdujo en la Enmienda 1 de la norma ISO C90.

Función: long long int strtoq (const char *restringir cadena, char **restringir tailptr, int base)

strtoq ("string-to-quad-word") es el nombre BSD para strtoll.

Función: long long int wcstoq (const wchar_t *restringir cadena, wchar_t **restringir punto de cola, entero base)

La función wcstoq es equivalente a la función strtoq en casi todos los aspectos, pero maneja cadenas de caracteres anchos.

La función wcstoq es una extensión de GNU.

Función: unsigned long long int strtoull (const char *restringir cadena, char **restringir tailptr, int base)

La función strtoull está relacionada con strtoll de la misma manera que strtoul está relacionada con strtol.

La función strtoull se introdujo en ISO C99.

Función: unsigned long long int wcstoull (const wchar_t *restringir cadena, wchar_t **restringir tailptr, int base)

La función wcstoull es equivalente a la función strtoull en casi todos los aspectos pero maneja cadenas de caracteres anchas.

La función wcstoull se introdujo en la Enmienda 1 de la norma ISO C90.

Función: unsigned long long int strtouq (const char *restringir cadena, char **restringir tailptr, base) entero

strtouq es el nombre BSD para strtoull.

Función: unsigned long long int wcstouq (const wchar_t *restringir cadena, wchar_t **restringir tailptr, int base)

La función wcstouq es equivalente a la función strtouq en casi todos los aspectos pero maneja cadenas de caracteres anchas.

La función wcstoq es una extensión de GNU.

Función: intmax_t strtoimax (const char *restringir cadena, char **restringir tailptr, int base)

La función strtoimax es como strtol excepto que devuelve un valor intmax_t y acepta números de un rango correspondiente.

Si la cadena tiene una sintaxis válida para un entero pero el valor no es representable Debido al desbordamiento, strtoimax devuelve INTMAX_MAX o INTMAX_MIN (consulte la sección números enteros), según corresponda para el signo del valor. También establece errno en ERANGE para Indicar que hubo desbordamiento.

Ver sección Números enteros Para una descripción del tipo intmax_t . El strtoimax Esta función se introdujo en la norma ISO C99.

Función: intmax_t wcstoimax (const wchar_t *restringir cadena, wchar_t **restringir punto de colar, entero base)

La función wcstoimax es equivalente a la función strtoimax en casi todos aspectos pero maneja cadenas de caracteres anchas.

La función wcstoimax se introdujo en ISO C99.

Función: uintmax_t strtoumax (const char *restringir cadena, char **restringir tailptr, base int)

La función strtoumax está relacionada con strtoimax de la misma manera que strtoul está relacionada. Para estrujarse.

Ver sección Números enteros Para una descripción del tipo intmax_t . El strtoumax Esta función se introdujo en la norma ISO C99.

Función: uintmax_t wcstoumax (const wchar_t *restringir cadena, wchar_t **restringir tailptr, int base)

La función wcstoumax es equivalente a la función strtoumax en casi todos aspectos pero maneja cadenas de caracteres anchas.

La función wcstoumax se introdujo en ISO C99.

Función: long int atol. Esta (const char *cadena)
función es similar a la función strtol con un argumento de 10, excepto que no necesita detectar base
errores de desbordamiento. La función atol se proporciona principalmente por compatibilidad con el código existente; usar strtol es más robusto.

Función: int atoi. Esta (const char *cadena)
función es similar a atol, excepto que devuelve un entero. La función atoi también se considera obsoleta; utilice strtol en su lugar.

Función: long long int atoll Esta función (const char *cadena)
es similar a atol, excepto que devuelve un long long int.

La función atoll se introdujo en la norma ISO C99. También está obsoleta (a pesar de haberse añadido recientemente); utilice strtoll en su lugar.

Ninguna de las funciones mencionadas en esta sección maneja representaciones alternativas de caracteres, como se describe en los datos de configuración regional. Algunas configuraciones regionales especifican el separador de miles y su uso, lo que facilita la lectura de números grandes. Para leer estos números, se deben usar las funciones scanf con el indicador `'' .

Aquí hay una función que analiza una cadena como una secuencia de números enteros y devuelve la suma de ellos:

```
int
suma_ints_de_cadena (carácter *cadena) {

    int suma = 0;

    mientras (1) { char
        *tail; int siguiente;


        /* Omite los espacios en blanco manualmente, para detectar el final. */ while (isspace
        (*string)) string++; if (*string == 0) break;



        /* Hay más espacios que no son blancos, */ /* por lo que
        debería ser otro número. */ errno = 0; /* Analizarlo. */ next = strtol
        (string, &tail,
        0); /* Agregarlo, si no se
        desborda. */ if (errno) printf ("Overflow\n"); else sum +=
        next; /* Avanzar más allá de él. */ string = tail;




    }

    devuelve suma;
}
```

Análisis de flotantes

Las funciones `str' se declaran en `stdlib.h' y las que empiezan por `wcs' se declaran en `wchar.h'. Cabe preguntarse sobre el uso de restrict en los prototipos de las funciones de esta sección. Parece inútil, pero el estándar ISO C lo utiliza (para las funciones definidas allí), así que también debemos hacerlo.

Función: doble strtod (const char *restringir cadena, char **restringir tailptr)

La función strtod ("cadena a doble") convierte la parte inicial de un número de cadena A un punto flotante, que se devuelve como un valor de tipo doble.

Esta función intenta descomponer cadena como sigue:

- Una secuencia (posiblemente vacía) de espacios en blanco. ¿Qué caracteres? Los espacios en blanco están determinados por la función isspace (ver sección Clasificación de personajes). Estos se descartan.
- Un signo más o menos opcional (`+' o `-').
- Un número de punto flotante en formato decimal o hexadecimal. El decimal El formato es:
  - Una secuencia de dígitos no vacía que opcionalmente contiene un punto decimal carácter, normalmente `.', pero depende de la configuración regional (ver sección Parámetros de formato numérico genérico).
  - Una parte de exponente opcional, que consiste en un carácter 'e' o 'E', un signo opcional y una secuencia de dígitos.

  El formato hexadecimal es el siguiente:
  - Un 0x o 0X seguido de una secuencia no vacía de dígitos hexadecimales opcionalmente contiene un carácter de punto decimal, normalmente '.', pero Depende de la configuración regional (ver sección Formato numérico genérico) Parámetros).
  - Una parte de exponente binario opcional, que consiste en un carácter 'p' o 'P', un signo opcional y una secuencia de dígitos.
- Cualquier carácter restante en la cadena. Si tailptr no es un puntero nulo, un El puntero a esta cola de la cadena se almacena en * tailptr .

Si la cadena está vacía, contiene solo espacios en blanco o no contiene una inicial subcadena que tiene la sintaxis esperada para un número de punto flotante, no Se realiza la conversión. En este caso, strtod devuelve un valor de cero y el valor devuelto en * tailptr es el valor de cadena.

En una configuración regional distinta a las configuraciones regionales estándar "C" o "POSIX" , esta función puede reconocer sintaxis adicional dependiente de la configuración regional.

Si la cadena tiene una sintaxis válida para un número de punto flotante pero el valor es fuera del rango de un doble, strtod indicará desbordamiento o subdesbordamiento como descrito en la sección Informe de errores mediante funciones matemáticas.

strtod reconoce cuatro cadenas de entrada especiales. Las cadenas "inf" e "infinity" son convertido a @math{@infinity{}}, o al valor representable más grande si el El formato de punto flotante no admite valores infinitos. Puede anteponer un "+" o un "-" a Especifica el signo. Se ignoran mayúsculas y minúsculas al escanear estas cadenas.

Las cadenas "nan" y "nan()" se convierten a NaN. Nuevamente, se distingue entre mayúsculas y minúsculas. personajes. Ignorado. Si se proporcionan, se utilizan de forma no especificada para seleccione una representación particular de NaN (puede haber varias).

Dado que cero es un resultado válido, así como el valor devuelto en caso de error, debe Verifique los errores de la misma manera que para strtol, examinando error y tailptr .

Función: float strtof (const char *string, char **tailptr)

Función: doble cadena larga (const char *string, char **tailptr)

Estas funciones son análogas a strtod, pero devuelven valores float y long double.

respectivamente. Informan errores de la misma manera que strtod. strtof puede ser sustancialmente más rápido que strtod, pero tiene menos precisión; por el contrario, strtold puede ser mucho más lento pero tiene más precisión (en sistemas donde long double es un tipo separado).

Estas funciones son extensiones de GNU y son nuevas en ISO C99.

Función: double wcstod Función: (const wchar_t *restringir cadena, wchar_t **restringir tailptr)
float wcstof Función: long (const wchar_t *cadena, wchar_t **tailptr)
double wcstold Las funciones wcstod, (const wchar_t *cadena, wchar_t **tailptr)
wcstof y wcstol son equivalentes en casi todos los aspectos a la Funciones strtod, strtof y strtold , pero maneja cadenas de caracteres anchas.

La función wcstod se introdujo en la Enmienda 1 de la norma ISO C90. La función wcstof y Las funciones wcstold se introdujeron en ISO C99.

Función: double atof Esta (const char *cadena)
función es similar a la función strtod , excepto que no necesita detectar Errores de desbordamiento y subdesbordamiento. La función atof se proporciona principalmente para... compatibilidad con el código existente; el uso de strtod es más robusto.

La biblioteca GNU C también proporciona versiones `_l' de estas funciones, que toman una argumento adicional, la configuración regional que se utilizará en la conversión. Consulte la sección Análisis de Números enteros.

Número a cadena del antiguo sistema V funciones

La antigua biblioteca System VC proporcionaba tres funciones para convertir números en cadenas, con semántica inusual y difícil de usar. La biblioteca GNU C también proporciona estas funciones y algunas extensiones naturales.

Estas funciones solo están disponibles en glibc y en sistemas descendientes de AT&T Unix. Por lo tanto, a menos que estas funciones hagan precisamente lo que necesitas, es mejor Utilice sprintf, que es estándar.

Todas estas funciones están definidas en 'stdlib.h'.

Función: char * ecvt (valor doble, int ndigit, int *decpt, int *neg)
La función ecvt convierte el número de punto flotante más valor a una cadena con at ndigit dígitos decimales. La cadena devuelta no contiene punto decimal ni signo. El primer dígito de la cadena no es cero (a menos que sea realmente cero) y valor El último dígito se redondea al más cercano. * se establece en el índice de la cadena de primer dígito después del punto decimal. * se establece en un valor distinto de cero si es valor es negativo, cero en caso contrario.

Si ndigit Los dígitos decimales excederían la precisión de un doble , se reduce a un es un valor específico del sistema.

La cadena devuelta se asigna estáticamente y se sobrescribe con cada llamada a ecvt.

Si valor es cero, se define por implementación si * departamento es 0 o 1.

Por ejemplo: ecvt (12.3, 5, &d, &n) devuelve "12300" y establece d a 2 y norte a 0.

Función: char * fcvt La (valor doble, int ndigit, int *decpt, int *neg)
función fcvt es como ecvt, pero especifica el número de dígitos después del ndigit
punto decimal. Si es menor que cero, se redondea al valor
@math{+1}'ésimo lugar a la izquierda del punto decimal. Por ejemplo, si
ndigit es -1, se redondeará a la decena más cercana. Si ndigit es negativo y
mayor que el número de dígitos a la izquierda del punto decimal se redondeará a un dígito valorvalor
significativo.

Si ndigit Los dígitos decimales excederían la precisión de un doble , se reduce a un
es un valor específico del sistema.

La cadena devuelta se asigna estáticamente y se sobrescribe con cada llamada a fcvt.

Función: char * gcvt (valor doble, int ndigit, char *buf)
gcvt es funcionalmente equivalente a `sprintf(buf, "%*g", ndigit, value'. Se proporciona
Solo por compatibilidad. Vuelve. buf .

Si ndigit Los dígitos decimales excederían la precisión de un doble , se reduce a un
es un valor específico del sistema.

Como extensiones, la biblioteca GNU C proporciona versiones de estas tres funciones que
tomar argumentos dobles y largos .

Función: char * qecvt Esta (valor doble largo, ndigit, *decpt, *neg)
entero entero entero
función es equivalente a ecvt excepto que toma un doble largo para la primera
parámetro y que está restringido por la precisión de un doble largo.

Función: char * qfcvt Esta (valor doble largo, int ndigit, int *decpt, int *neg)
función es equivalente a fcvt excepto que toma un doble largo para la primera
parámetro y que está restringido por la precisión de un doble largo.

Función: char * qgcvt Esta (valor doble largo, ndigit, char *buf)
entero
función es equivalente a gcvt excepto que toma un doble largo para la primera
parámetro y que está restringido por la precisión de un doble largo.

Las funciones ecvt y fcvt , y sus equivalentes dobles largos , devuelven una cadena
Ubicado en un búfer estático que se sobrescribe con la siguiente llamada a la función.
La biblioteca GNU C proporciona otro conjunto de funciones extendidas que escriben el
Cadena convertida en un búfer proporcionado por el usuario. Estos tienen el sufijo convencional _r .

gcvt_r no es necesario, porque gcvt ya utiliza un búfer proporcionado por el usuario.

Función: char * ecvt_r (valor doble, ndigit, *decpt, char *buf,
entero entero entero *neg,
tamaño_t len)
La función ecvt_r es la misma que ecvt, excepto que coloca su resultado en el
búfer especificado por el usuario al que apunta con longitud buf lente.

Esta función es una extensión GNU.

Función: char * fcvt_r (valor doble, int ndigit, int *decpt, int *neg, char *buf,
talla_t len)
La función fcvt_r es la misma que fcvt, excepto que coloca su resultado en el
búfer especificado por el usuario al que apunta con longitud buf lente.

Esta función es una extensión GNU.

Función: char * qecvt_r (valor doble largo, int ndigit, int *decpt, int *neg, char *buf, len)
talla_t

La función qecvt_r es la misma que qecvt, excepto que coloca su resultado en el búfer especificado por el usuario al que apunta con longitud buf lente.

Esta función es una extensión GNU.

Función: char * qfcvt_r (valor doble largo, ndigit, *decpt, entero entero entero *neg, char *buf, tamaño_t len)

La función qfcvt_r es la misma que qfcvt, excepto que coloca su resultado en el búfer especificado por el usuario al que apunta con longitud buf lente.

Esta función es una extensión GNU.

Ve al primero, anterior, próximo, último sección, tabla de contenidos.