

Appendice E

Funzioni della libreria standard

Questa appendice descrive tutte le funzioni di libreria supportate dal C89 e dal C99¹. Quando utilizzate questa appendice tenete a mente le seguenti cose:

- Per questioni di brevità e chiarezza sono stati omessi molti dettagli. Alcune funzioni (in particolare la `printf`, la `scanf` e le loro varianti) vengono trattate nel dettaglio in altre parti del libro, di conseguenza la loro descrizione qui è minimale. Per maggiori informazioni su una funzione (inclusi degli esempi su come questa viene usata), leggete la sezione o le sezioni elencate in corsivo nell'angolo in basso a sinistra della descrizione.
- Come in altre parti del libro, il *corsivo* viene usato per indicare le differenze del C99. I nomi e i prototipi delle funzioni che sono state aggiunte nel C99 sono in corsivo. Le modifiche ai prototipi C89 (l'aggiunta della parola `restrict` alla dichiarazione di certe funzioni) sono anch'esse scritte in corsivo.
- In questa appendice sono incluse le macro parametriche (ad eccezione delle macro per tipi generici presenti in `<tgmath.h>`). Ogni prototipo per una macro è seguito dalla parola *macro*.
- Nel C99 alcune funzioni `<math.h>` possiedono tre versioni (una per il tipo `float`, una per il `double` e una per il `long double`). Tutte e tre sono raggruppate in una singola voce sotto il nome della versione `double`. Per esempio: c'è una sola voce per le funzioni `acos`, `acosf` e `acosl` sotto il nome `acos`. Il nome di ogni versione aggiuntiva (`acosf` e `acosl` nel nostro esempio) compaiono a sinistra del suo prototipo. Le funzioni `<complex.h>`, le quali sono a loro volta presenti in tre versioni, sono trattate in modo simile.
- La maggior parte delle funzioni `<wchar.h>` sono le versioni wide character delle funzioni che si trovano negli altri header. A meno che non ci sia un comportamento significativamente diverso, la descrizione di ogni funzione wide character rimanda semplicemente il lettore alla corrispondente funzione che si trova altrove.
- Se qualche aspetto del comportamento della funzione viene descritto come *definito dall'implementazione*, questo significa che dipende da com'è implementata la libreria C. La funzione si comporterà sempre in modo consistente, ma il risultato potrà variare da un sistema a un altro (in altre parole, controllate il manuale per vedere cosa succede). Un comportamento *indefinito* invece è una cattiva notizia: non solo il comportamento varia tra i diversi sistemi, ma il programma può comportarsi in modo strano o persino andare in crash.
- Le descrizioni di molte funzioni `<math.h>` si riferiscono ai termini *errore di dominio* ed *errore di intervallo*. Il modo nel quale questi errori vengono segnalati è cambiato dal C89 al C99. Per capire come il C89 tratti questi errori leggete la Sezione 23.3. Per leggere invece come il C99 tratti questi errori andate alla Sezione 23.4.
- Il comportamento delle seguenti funzioni è affetto dalla localizzazione corrente:

<code><ctype.h></code>	Tutte le funzioni.
<code><stdio.h></code>	Funzioni per l'input/output formattato.
<code><stdlib.h></code>	Funzioni di conversione tra caratteri multibyte e wide character, funzioni di conversione numerica.
<code><string.h></code>	<code>strcoll</code> , <code>strxfrm</code> .
<code><time.h></code>	<code>strftime</code> .
<code><wchar.h></code>	<code>wscoll</code> , <code>wcsftime</code> , <code>wcsxfrm</code> , funzioni per l'input/output formattato, funzioni di conversione numerica, funzioni di conversione tra i multibyte estesi e i wide character.
<code><wctype.h></code>	Tutte le funzioni.

Per esempio: di solito la funzione `isalpha` controlla se un carattere risiede tra `a` e `z` o tra `A` e `Z`. In alcune localizzazioni, anche altri caratteri sono considerati alfabetici.

abort *Terminare il programma*

`<stdlib.h>`

```
void abort(void);
```

Genera il segnale `SIGABRT`. Se il segnale non viene intercettato (o se il gestore del segnale termina), il programma termina in modo anormale e restituisce un codice definito dall'implementazione indicante una terminazione senza successo. Se i buffer di output vengano svuotati, gli stream aperti vengano chiusi o se i file temporanei vengano rimossi dipende dall'implementazione.

¹ Questo materiale è tratto dallo standard ISO/IEC 9899:1999.

abs *Valore assoluto intero*

<stdlib.h>

```
int abs(int j);
```

Restituisce

Il valore assoluto di j . Il comportamento non è definito se il valore assoluto di j non può essere rappresentato

26.2

acos *Arcocoseno*

<math.h>

```
double acos(double x);
```

```
float acosf(float x);
```

acosf**acosl**

```
long double acosl(long double x);
```

Restituisce

L'arcocoseno di x , il valore restituito è compreso nell'intervallo da 0 a π . Se x non è compreso tra -1 e +1 si verifica un errore di dominio.

23.3

acosh *Arcocoseno iperbolico (C99)*

<math.h>

```
double acosh(double x);
```

```
float acoshf(float x);
```

acoshf**acoshl**

```
long double acoshl(long double x);
```

Restituisce

L'arcocoseno iperbolico di x , il valore restituito appartiene all'intervallo da 0 a $+\infty$. Se x è minore di 1 si verifica un errore di dominio.

23.4

asctime *Converte un'ora di tipo broken-down in una stringa*

<time.h>

```
char *asctime(const struct tm *timeptr);
```

Restituisce

Un puntatore a una stringa terminante con il carattere null della forma

```
Sun Jun 3 17:48:34 2007\n
```

costruita a partire dall'ora broken-down presente nella struttura puntata da `timeptr`.

26.3

asin *Arcoseno*

<math.h>

```
double asin(double x);
```

```
float asinf(float x);
```

asinf**asinl**

```
long double asinl(long double x);
```

Restituisce

L'arcoseno di x . Il valore restituito appartiene all'intervallo da $-\pi/2$ a $+\pi/2$. Se x non è compreso tra -1 e +1 si verifica un errore di dominio.

23.3

asinh *Arcoseno iperbolico (C99)*

<math.h>

```
double asinh(double x);
```

```
float asinhf(float x);
```

asinhf**asinhl**

```
long double asinhl(long double x);
```

Restituisce

L'arcoseno iperbolico di x .

23.4

assert *Verifica se un'espressione è vera*

<assert.h>

```
void assert(scalar expression);
```

macro

Se il valore di `expression` è diverso da zero, la funzione non fa nulla. Se il valore è uguale a zero, la `assert` scrive un messaggio su `stderr` (specificando il testo di `expression`, il nome del file sorgente contenete la chiamata e il

numero di riga nella quale si trova), successivamente termina il programma chiamando la funzione `abort`. Per disabilitare la `assert`, definite la macro `NDEBUG` prima di includere l'header `<assert.h>`. *Modifiche del C99*: è ammesso che l'argomento sia di un qualsiasi tipo scalare, mentre il C89 specifica che il tipo debba essere `int`. Inoltre il C99 richiede che il messaggio scritto dalla `assert` includa il nome della funzione nella quale compare la `assert`, mentre il C89 non presenta quest'obbligo. 24.1

atan *Arcotangente* <math.h>

```
double atan(double x);
```

atanf

```
float atanf(float x);
```

atanl

```
long double atanl(long double x);
```

Restituisce
L'arcotangente di x . Il valore restituito appartiene all'intervallo da $-\pi/2$ a $+\pi/2$. 23.3

atan2 *Arcotangente del quoziente* <math.h>

```
double atan2(double y, double x);
```

atan2f

```
float atan2f(float y, float x);
```

atan2l

```
long double atan2l(long double y, long double x);
```

Restituisce
L'arcotangente di y/x . Il valore restituito è compreso nell'intervallo da $-\pi$ a $+\pi$. Se x e y sono entrambi uguali a zero si verifica un errore di dominio. 23.3

atanh *Arcotangente iperbolica (C99)* <math.h>

```
double atanh(double x);
```

atanhf

```
float atanhf(float x);
```

atanhl

```
long double atanh1(long double x);
```

Restituisce
L'arcotangente iperbolica di x . Se x non è compreso tra -1 e $+1$ si verifica un errore di dominio. Se x è uguale a -1 o a $+1$ si può verificare un errore di intervallo. 23.4

atexit *Registrazione di una funzione da chiamare all'uscita del programma* <stdlib.h>

```
int atexit(void (*func)(void));
```

Registra come funzione di terminazione la funzione puntata da `func`. La funzione verrà chiamata se il programma termina normalmente (attraverso `return` o `exit` ma non `abort`). *Restituisce*

Zero se ha successo, un valore diverso da zero se non va a buon fine (è stato raggiunto un limite dipendente dall'implementazione). 26.2

atof *Conversione di una stringa in un numero a virgola mobile* <stdlib.h>

```
double atof(const char *nptr);
```

Restituisce
Un valore `double` corrispondente alla più lunga parte iniziale della stringa puntata da `nptr` che abbia la forma di un numero a virgola mobile. Se la conversione non può essere eseguita restituisce il valore zero. Il comportamento della funzione non è definito se il numero non può essere rappresentato. 26.2

atoi *Conversione di una stringa in un intero* <stdlib.h>

```
int atoi(const char *nptr);
```

Restituisce
Un valore `int` corrispondente alla più lunga parte iniziale della stringa puntata da `nptr` che abbia la forma di un intero. Se la conversione non può essere eseguita restituisce il valore zero. Il comportamento della funzione non è definito se il numero non può essere rappresentato. 26.2

atol *Conversione di una stringa in un long int* <stdlib.h>

```
long int atol(const char *nptr);
```

Restituisce

Un valore `long int` corrispondente alla più lunga parte iniziale della stringa puntata da `nptr` che abbia la forma di un intero. Se la conversione non può essere effettuata restituisce il valore zero. Il comportamento della funzione non è definito se il numero non può essere rappresentato. 26.2

`atoll` *Conversione di una stringa in un long long int (C99)* <stdlib.h>

```
long long int atoll(const char *nptr);
```

Restituisce

Un valore `long long int` corrispondente alla più lunga parte iniziale della stringa puntata da `nptr` che abbia la forma di un intero. Se la conversione non può essere effettuata restituisce il valore zero. Il comportamento della funzione non è definito se il numero non può essere rappresentato. 26.2

`bsearch` *Ricerca binaria* <stdlib.h>

```
void *bsearch(const void *key, const void *base,
              size_t memb, size_t size,
              int (*compar)(const void *,
                           const void *));
```

Ricerca il valore puntato da `key` nel vettore ordinato puntato da `base`. Il vettore possiede `nmemb` elementi, ognuno lungo `size` byte. `compar` è un puntatore a una funzione di confronto. Quando le vengono passati i puntatori alla chiave e a un elemento del vettore (in quell'ordine) la funzione di confronto deve restituire un intero negativo, pari a zero o positivo a seconda che la chiave sia minore, uguale o maggiore dell'elemento.

Restituisce

Un puntatore a un elemento del vettore che è uguale alla chiave. Se la chiave non viene trovata restituisce un puntatore nullo. 26.2

`btowc` *Conversione di un byte in un wide character (C99)* <wchar.h>

```
wint_t btowc(int c);
```

Restituisce

La rappresentazione `wide character` di `c`. Restituisce `WEOF` se `c` è uguale a `EOF` o se `c` (quando viene fatto un cast al tipo `unsigned char`) non è un carattere a singolo byte valido nello stato di shift iniziale. 25.5

`cabs` *Valore assoluto complesso (C99)* <complex.h>

```
double cabs(double complex z);
```

`cabsf`

```
float cabsf(float complex z);
```

`cabsl`

```
long double cabsl(long double complex z);
```

Restituisce

Il valore assoluto complesso di `z`. 27.4

`acos` *Arcocoseno (C99)* <complex.h>

```
double complex acos(double complex z);
```

`acosf`

```
float complex acosf(float complex z);
```

`acosl`

```
long double complex acosl(long double complex z);
```

Restituisce

L'arcocoseno complesso di `z`, con dei punti di diramazione al di fuori dell'intervallo $[-1, +1]$ lungo l'asse reale. Il valore restituito risiede nella striscia priva di confini lungo l'asse immaginario e delimitata dall'intervallo $[0, \pi]$ lungo l'asse reale. 27.4

`cacosh` *Arcocoseno iperbolico complesso (C99)* <complex.h>

```
double complex cacosh(double complex z);
```

`cacoshf`

```
float complex cacoshf(float complex z);
```

`cacoshl`

```
long double complex cacoshl(long double complex z);
```

Restituisce

L'arcocoseno iperbolico complesso di `z`, con un punto di diramazione ai valori minori di 1 lungo l'asse reale. Il valore restituito risiede nella semistriscia dei valori non negativi lungo l'asse reale e nell'intervallo $[-i\pi, +i\pi]$ lungo l'asse

immaginario.	27.4
calloc <i>Allocazione e azzeramento di un blocco di memoria</i>	<stdlib.h>
void *calloc(size_t nmemb, size_t size);	
Alloca un blocco di memoria per un vettore con nmemb elementi, ognuno di size byte. Il blocco viene azzerato imponendo tutti i bit a zero.	
<i>Restituisce</i>	
Un puntatore all'inizio del blocco. Se non può essere allocato un blocco della dimensione richiesta restituisce un puntatore nullo.	17.3
carg <i>Argomento complesso (C99)</i>	<complex.h>
double carg(double complex z);	
float cargf(float complex z);	cargf
long double cargl(long double complex z);	cargl
<i>Restituisce</i>	
L'argomento (angolo di fase) di z, con un punto di diramazione lungo l'asse immaginario negativo. Il valore restituito risiede nell'intervallo $[-\pi, +\pi]$.	27.4
casin <i>Arcoseno complesso (C99)</i>	<complex.h>
double complex casin(double complex z);	
float complex casinf(float complex z);	casinf
long double complex casinl(long double complex z);	casinl
<i>Restituisce</i>	
L'arcoseno complesso di z, con dei punti di diramazione al di fuori dell'intervallo $[-1, +1]$ lungo l'asse reale. Il valore restituito risiede su una striscia priva di confini lungo l'asse immaginario e delimitata dall'intervallo $[-\pi/2, +\pi/2]$ lungo l'asse reale.	27.4
casinh <i>Arcoseno iperbolico complesso (C99)</i>	<complex.h>
double complex casinh(double complex z);	
float complex casinhf(float complex z);	casinhf
long double complex casinhl(long double complex z);	casinhl
<i>Restituisce</i>	
L'arcoseno iperbolico complesso di z, con dei punti di diramazione al di fuori dell'intervallo $[-i, +i]$ lungo l'asse immaginario. Il valore restituito risiede su una striscia priva di confini lungo l'asse reale e delimitata dall'intervallo $[-i\pi/2, +i\pi/2]$ lungo l'asse immaginario.	27.4
catan <i>Arcotangente complessa (C99)</i>	<complex.h>
double complex catan(double complex z);	
float complex catanf(float complex z);	catanf
long double complex catanl(long double complex z);	catanl
<i>Restituisce</i>	
L'arcotangente complessa di z, con dei punti di diramazione al di fuori dell'intervallo $[-i, +i]$ lungo l'asse immaginario. Il valore restituito risiede su una striscia priva di confini lungo l'asse immaginario e delimitata dall'intervallo $[-\pi/2, +\pi/2]$ lungo l'asse reale.	27.4
catanh <i>Arcotangente iperbolica complessa (C99)</i>	<complex.h>
double complex catanh(double complex z);	
float complex catanhf(float complex z);	catanhf
long double complex catanhl(long double complex z);	catanhl
<i>Restituisce</i>	

L'arcotangente iperbolica complessa di z , con dei punti di diramazione al di fuori dell'intervallo $[-1, +1]$ lungo l'asse reale. Il valore restituito risiede su una striscia priva di confini lungo l'asse reale e delimitata dall'intervallo $[-i\pi/2, +i\pi/2]$ lungo l'asse immaginario. 27.4

`cbrt` *Radice cubica (C99)* <math.h>

`double cbrt(double x);` **`cbrtf`**

`float cbrtf(float x);` **`cbrtl`**

`long double cbrtl(long double x);`

Restituisce
La radice cubica reale di x . 23.4

`ccos` *Coseno complesso (C99)* <complex.h>

`double complex ccos(double complex z);` **`ccosf`**

`float complex ccosf(float complex z);` **`ccosl`**

`long double complex ccosl(long double complex z);`

Restituisce
Il coseno complesso di z . 27.4

`ccosh` *Coseno iperbolico complesso (C99)* <complex.h>

`double complex ccosh(double complex z);` **`ccoshf`**

`float complex ccoshf(float complex z);` **`ccoshl`**

`long double complex ccoshl(long double complex z);`

Restituisce
Il coseno iperbolico complesso di z . 27.4

`ceil` *Ceiling* <math.h>

`double ceil(double x);` **`ceilf`**

`float ceilf(float x);` **`ceill`**

`long double ceill(long double x);`

Restituisce
Il più piccolo intero maggiore o uguale a x . 23.3

`cexp` *Esponenziale complesso con base e (C99)* <complex.h>

`double complex cexp(double complex z);` **`cexpf`**

`float complex cexpf(float complex z);` **`cexpl`**

`long double complex cexpl(long double complex z);`

Restituisce
L'esponenziale complesso con base e di z . 27.4

`cimag` *Parte immaginaria di un numero complesso (C99)* <complex.h>

`double cimag(double complex z);` **`cimagf`**

`float cimagf(float complex z);` **`cimagl`**

`long double cimagl(long double complex z);`

Restituisce
La parte immaginaria di z . 27.4

`clearerr` *Azzeramento degli errori di uno stream* <stdio.h>

`void clearerr(FILE *stream);`

 Azzerare gli indicatori di end-of-file e di errore dello stream puntato da `stream`.

22.3

clock *Clock del processore*

<time.h>

`clock_t clock(void);`*Restituisce*

Il tempo di processore trascorso (misurato in “*clock ticks*”) a partire dall’inizio dell’esecuzione del programma (per convertirlo in secondi va diviso per `CLOCKS_PER_SEC`). Restituisce (`clock_t`) (-1) se il tempo non è disponibile o se non può essere rappresentato.

26.3

clog *Logaritmo naturale complesso (C99)*

<complex.h>

`double complex clog(double complex z);``float complex clogf(float complex z);``long double complex clogl(long double complex z);`*Restituisce*

Il logaritmo naturale (base e) di z , con un punto di diramazione lungo l’asse reale negativo. Il valore restituito risiede su una striscia priva di confini lungo l’asse reale e delimitata dall’intervallo $[-i\pi, +i\pi]$ lungo l’asse immaginario.

27.4

conj *Complesso coniugato (C99)*

<complex.h>

`double complex conj(double complex z);``float complex conjf(float complex z);``long double complex conjl(long double complex z);`*Restituisce*Il complesso coniugato di z .

27.4

copysign *Copia del segno (C99)*

<math.h>

`double copysign(double x, double y);``float copysignf(float x, float y);``long double copysignl(long double x, long double y);`*Restituisce*Un valore con il valore assoluto di x e il segno di y .

23.4

cos *Coseno*

<math.h>

`double cos(double x);``float cosf(float x);``long double cosl(long double x);`*Restituisce*Il coseno di x (misurato in radianti).

23.3

cosh *Coseno iperbolico*

<math.h>

`double cosh(double x);``float coshf(float x);``long double coshl(long double x);`*Restituisce*Il coseno iperbolico di x . Se la magnitudine di x è troppo grande si verifica un errore di intervallo.

23.3

cpow *Potenza complessa (C99)*

<complex.h>

`double complex cpow(double complex x,
double complex y);``float complex cpowf(float complex x,`**cpowf**

<code>float complex y);</code>		
<code>long double complex cpowl(long double complex x, long double complex y);</code>		cpowl
<i>Restituisce</i> x elevato alla potenza y, con un punto di ramificazione per il primo parametro lungo la porzione negativa dell'asse reale.		27.4
<hr/>		
cproj <i>Proiezione complessa (C99)</i>	<complex.h>	
<code>double complex cproj(double complex z);</code>		
<code>float complex cprojf(float complex z);</code>		cprojf
<code>long double complex cprojl(long double complex z);</code>		cprojl
<i>Restituisce</i> La proiezione di z sulla sfera di Riemann. Viene restituito z a meno che una delle sue parti non sia infinita, in tal caso il valore restituito è INFINITY + I * copysign(0.0, cimag(z)).		27.4
<hr/>		
creal <i>Parte reale di un numero complesso (C99)</i>	<complex.h>	
<code>double creal(double complex z);</code>		
<code>float crealf(float complex z);</code>		crealf
<code>long double creall(long double complex z);</code>		creall
<i>Restituisce</i> La parte reale di z.		27.4
<hr/>		
csin <i>Seno complesso (C99)</i>	<complex.h>	
<code>double complex csin(double complex z);</code>		
<code>float complex csinf(float complex z);</code>		csinf
<code>long double complex csinl(long double complex z);</code>		csinl
<i>Restituisce</i> Il seno complesso di z.		27.4
<hr/>		
csinh <i>Seno iperbolico complesso (C99)</i>	<complex.h>	
<code>double complex csinh(double complex z);</code>		
<code>float complex csinhf(float complex z);</code>		csinhf
<code>long double complex csinhl(long double complex z);</code>		csinhl
<i>Restituisce</i> Il seno iperbolico complesso di z.		27.4
<hr/>		
csqrt <i>Radice quadrata complessa (C99)</i>	<complex.h>	
<code>double complex csqrt(double complex z);</code>		
<code>float complex csqrtf(float complex z);</code>		csqrtf
<code>long double complex csqrtl(long double complex z);</code>		csqrtl
<i>Restituisce</i> La radice quadrata complessa di z, con un punto di diramazione lungo la parte negativa dell'asse reale.		27.4
<hr/>		
ctan <i>Tangente complessa (C99)</i>	<complex.h>	
<code>double complex ctan(double complex z);</code>		
<code>float complex ctanf(float complex z);</code>		ctanf
		ctanl

```
long double complex ctanl(long double complex z);
```

Restituisce

La tangente complessa di z .

27.4

ctanh *Tangente iperbolica complessa (C99)*

<complex.h>

```
double complex ctanh(double complex z);
```

ctanhf

```
float complex ctanhf(float complex z);
```

ctanhl

```
long double complex ctanh1(long double complex z);
```

Restituisce

La tangente iperbolica complessa di z .

27.4

ctime *Conversione di un'ora di tipo calendar time in una stringa*

<time.h>

```
char *ctime(const time_t *timer);
```

Restituisce

Un puntatore a una stringa che descrive l'ora locale equivalente all'ora del formato calendar time puntata da `timer`. È equivalente a `asctime(localtime(timer))`.

26.3

difftime *Differenza tra ore*

<time.h>

```
double difftime(time_t time1, time_t time0);
```

Restituisce

La differenza misurata in secondi tra `time0` (un'orario precedente) e `time1`.

26.3

div *Divisione intera*

<stdlib.h>

```
div_t div(int numer, int denom);
```

Restituisce

Una struttura `div_t` contenete dei membri chiamati `quot` (il quoziente ottenuto dividendo `numer` per `denom`) e `rem` (il resto). Il comportamento non è definito se una delle due parti del risultato non può essere rappresentata.

26.2

erf *Funzione di errore (C99)*

<math.h>

```
double erf(double x);
```

erff

```
float erff(float x);
```

erfl

```
long double erfl(long double x);
```

Restituisce

$erf(x)$, dove erf è la funzione di errore gaussiana.

23.4

erfc *Funzione di errore complementare (C99)*

<math.h>

```
double erfc(double x);
```

erfcf

```
float erfcf(float x);
```

erfcl

```
long double erfcl(long double x);
```

Restituisce

$erfc(x) = 1 - erf(x)$, dove erf è la funzione di errore gaussiana. Se x è troppo grande si verifica un errore di intervallo.

23.4

exit *Uscita dal programma*

<stdlib.h>

```
void exit(int status);
```

Chiama tutte le funzioni registrate con la `atexit`, svuota tutti i buffer di output, chiude tutti gli stream aperti, rimuove tutti i file creati dalla `tmpfile` e termina il programma. Il valore di `status` indica se il programma è terminato normalmente. Gli unici valori portabili per `status` sono 0 ed `EXIT_SUCCESS` (entrambi indicano una terminazione con successo), oltre che `EXIT_FAILURE` (terminazione senza successo).

9.5, 26.2

_Exit *Uscita dal programma (C99)*

<stdlib.h>

```
void _Exit(int status);
```

Provoca una terminazione normale del programma. Non chiama le funzioni registrate con la `atexit` o gli handler dei segnali registrati con la `signal`. Lo stato restituito viene determinato nello stesso modo usato dalla `exit`. Se i buffer di output vengano svuotati, se gli stream aperti vengano chiusi o se dei file temporanei vengano rimossi dipende dall'implementazione. 26.2

exp *Esponenziale in base e* <math.h>
`double exp(double x);`
`float expf(float x);` **expf**
`long double expl(long double x);` **expl**
Restituisce
e elevato alla potenza *x*. Se il valore assoluto di *x* è troppo grande si verifica un errore di intervallo. 23.3

exp2 *Esponenziale in base 2 (C99)* <math.h>
`double exp2(double x);` **exp2f**
`float exp2f(float x);` **exp2l**
`long double exp2l(long double x);`
Restituisce
2 elevato alla potenza *x*. Se il valore assoluto di *x* è troppo grande si verifica un errore di intervallo. 23.4

expm1 *Esponenziale in base e meno 1 (C99)* <math.h>
`double expm1(double x);` **expm1f**
`float expm1f(float x);` **expm1l**
`long double expm1l(long double x);`
Restituisce
e elevato alla potenza *x*, meno 1. Se il valore assoluto di *x* è troppo grande si verifica un errore di intervallo. 23.4

fabs *Valore assoluto in virgola mobile* <math.h>
`double fabs(double x);` **fabsf**
`float fabsf(float x);` **fabsl**
`long double fabsl(long double x);`
Restituisce
Valore assoluto di *x*. 23.3

fclose *Chiusura di un file* <stdio.h>
`int fclose(FILE *stream);`
Chiude lo stream puntato da `stream`. Svuota qualsiasi output non scritto che è rimasto nel buffer dello stream. Dealloca il buffer se questo è stato allocato automaticamente.
Restituisce
Zero se ha successo, EOF se è stato rilevato un errore. 22.2

fdim *Differenza positiva (C99)* <math.h>
`double fdim(double x, double y);` **fdimf**
`float fdimf(float x, float y);` **fdiml**
`long double fdiml(long double x, long double y);`
Restituisce

$$\begin{cases} x - y & \text{if } x > y \\ +0 & \text{if } x \leq y \end{cases}$$
Differenza positivo di *x* e *y*:
Può verificarsi un errore di intervallo. 23.4

feclearexcept *Azzeramento delle eccezioni floating point (C99)* <fenv.h>

```
int feclearexcept(int excepts);
```

Cerca di azzerare le eccezioni floating point rappresentate da `excepts`.

Restituisce

Il valore zero se `excepts` è uguale a zero o se tutte le eccezioni specificate sono state azzerate con successo. Negli altri casi restituisce un valore diverso da zero. 27.6

fegetenv *Ricavare l'ambiente floating point (C99)* <fenv.h>

```
int fegetenv(fenv_t *envp);
```

Cerca di salvare l'ambiente floating point corrente nell'oggetto puntato da `envp`.

Restituisce

Zero se l'ambiente è stato salvato con successo, altrimenti restituisce un valore diverso da zero. 27.6

fegetexceptflag *Ricavare i flag di eccezione floating point (C99)* <fenv.h>

```
int fegetexceptflag(fexcept_t *flagp, int excepts);
```

Cerca di recuperare lo stato dei flag di stato floating point rappresentato da `excepts` e li salva nell'oggetto puntato da `flagp`.

Restituisce

Il valore zero se lo stato di tutti i flag è stato salvato con successo, altrimenti restituisce un valore diverso da zero. 27.6

fegetround *Ricavare la direzione di arrotondamento floating point (C99)* <fenv.h>

```
int fegetround(void);
```

Restituisce

Il valore della macro che rappresenta la direzione di arrotondamento corrente. Restituisce un valore negativo se la direzione di arrotondamento corrente non può essere determinata o se non corrisponde a nessuna macro per la direzione di arrotondamento. 27.6

feholdexcept *Salvataggio dell'ambiente floating point (C99)* <fenv.h>

```
int feholdexcept(fenv_t *envp);
```

Salva l'ambiente floating point corrente nell'oggetto puntato da `envp`, azzeri i flag di stato floating point e cerca di installare una modalità non-stop per tutte le eccezioni floating point.

Restituisce

Il valore zero se un'eccezione floating point di tipo non-stop è stata installata con successo, altrimenti restituisce un valore diverso da zero. 27.6

feof *Test per end-of-file* <stdio.h>

```
int feof(FILE *stream);
```

Restituisce

Un valore diverso da zero se per lo stream puntato da `stream` è impostato l'indicatore di fine file, altrimenti restituisce il valore zero. 22.3

feraiseexcept *Generazione di eccezioni floating point (C99)* <fenv.h>

```
int feraiseexcept(int excepts);
```

Cerca di generare le eccezioni floating point supportate rappresentate da `excepts`.

Restituisce

Il valore zero se `excepts` è uguale a zero e se tutte le eccezioni specificate sono state sollevate con successo. Negli altri casi restituisce un valore diverso da zero. 27.6

ferror *Test per l'indicatore di errore* <stdio.h>

```
int ferror(FILE *stream);
```

Restituisce

Un valore diverso da zero se per lo stream puntato da `stream` è impostato l'indicatore di errore, altrimenti restituisce il valore zero. 22.3

fesetenv *Impostare l'ambiente floating point (C99)* <fenv.h>

```
int fesetenv(const fenv_t *envp);
```

Cerca di impostare l'ambiente floating point rappresentato dall'oggetto puntato da `envp`.

Restituisce

Il valore zero se l'ambiente è stato impostato con successo, altrimenti restituisce un valore diverso da zero. 27.6

`fesetexceptflag` *Impostare i flag per le eccezioni floating point (C99)* <fenv.h>

```
int fesetexceptflag(const fexcept_t *flagp,
                   int excepts);
```

Cerca di impostare i flag di stato floating point rappresentati da `excepts` allo stato contenuto nell'oggetto puntato da `flagp`.

Restituisce

Il valore zero se `excepts` è uguale a zero o se tutte le eccezioni specificate non state impostate con successo, altrimenti restituisce un valore diverso da zero. 27.6

`fesetround` *Impostare la direzione di arrotondamento floating point (C99)* <fenv.h>

```
int fesetround(int round);
```

Cerca di imporre la direzione di arrotondamento rappresentata da `round`.

Restituisce

Il valore zero se la direzione di arrotondamento richiesta è stata impostata, altrimenti restituisce un valore diverso da zero. 27.6

`fetestexcept` *Impostare i flag per le eccezioni floating point (C99)* <fenv.h>

```
int fetestexcept(int excepts);
```

Restituisce

L'or bitwise delle macro per le eccezioni floating point corrispondenti ai flag correntemente impostati per le eccezioni rappresentate da `excepts`. 27.6

`feupdateenv` *Aggiornamento dell'ambiente floating point (C99)* <fenv.h>

```
int feupdateenv(const fenv_t *envp);
```

Cerca di salvare le eccezioni floating point che sono correntemente sollevate, installa l'ambiente floating point rappresentato dall'oggetto puntato da `envp` e poi solleva le eccezioni salvate.

Restituisce

Il valore zero se tutte le azioni sono state portate a termine con successo, altrimenti restituisce un valore diverso da zero. 27.6

`fflush` *Svuotare il buffer del file* <stdio.h>

```
int fflush(FILE *stream);
```

Scrive tutti i dati non scritti presenti nel buffer associato a `stream`, il quale punta a uno stream che era stato aperto per l'output o per l'aggiornamento. Se `stream` è un puntatore nullo, la `fflush` svuota tutti gli stream che hanno nel buffer dei dati non scritti.

Restituisce

Il valore zero se l'operazione ha successo, il valore EOF se si verifica un errore di scrittura. 22.2

`fgetc` *Lettura di un carattere da file* <stdio.h>

```
int fgetc(FILE *stream);
```

Legge un carattere dallo stream puntato da `stream`.

Restituisce

Il carattere letto dallo stream. Se la `fgetc` incontra la fine dello stream, imposta l'indicatore di end-of-file e restituisce il valore EOF. Se si verifica un errore di lettura, la `fgetc` imposta l'indicatore di errore dello stream e restituisce il valore EOF. 22.4

`fgetpos` *Ottenere la posizione del file* <stdio.h>

```
int fgetpos(FILE * restrict stream,
            fpos_t * restrict pos);
```

Salva la posizione corrente se lo stream puntato da `stream` nell'oggetto puntato da `pos`.

Restituisce

Il valore zero se è ha successo. Se la chiamata fallisce, restituisce un valore diverso da zero e salva nella variabile

errno un valore positivo definito dall'implementazione.

22.7

fgets *Lettura di una stringa da file*

<stdio.h>

```
char *fgets(char * restrict s, int n,
            FILE * restrict stream);
```

Legge dei caratteri dallo stream puntato da `stream` e li salva nel vettore puntato da `s`. La lettura si interrompe: al primo carattere new-line (il quale viene salvato nella stringa), quando $n - 1$ caratteri sono stati letti o alla fine del file. La `fgets` aggiunge il carattere null alla fine della stringa.

Restituisce

`s` (un puntatore al vettore nel quale viene salvato l'input). Restituisce un puntatore nullo se si verifica un errore nella lettura o se incontra la fine del file prima di aver salvato qualsiasi carattere.

22.5

fgetwc *Lettura di un wide character da file (C99)*

<wchar.h>

```
wint_t fgetwc(FILE *stream);
```

La versione wide character della `fgetc`.

25.5

fgetws *Lettura di una stringa wide da file (C99)*

<wchar.h>

```
wchar_t *fgetws(wchar_t * restrict s, int n,
                FILE * restrict stream);
```

La versione wide character della `fgets`.

25.5

floor *Floor*

<math.h>

```
double floor(double x);
```

floorf

```
float floorf(float x);
```

floorl

```
long double floorl(long double x);
```

Restituisce

Il più grande intero minore o uguale a x .

23.3

fma *Moltiplicazione e somma in virgola mobile (C99)*

<math.h>

```
double fma(double x, double y, double z);
```

fmaf

```
float fmaf(float x, float y, float z);
```

fmal

```
long double fmal(long double x, long double y,
                 long double z);
```

Restituisce

Il valore $(x \times y) + z$. Il risultato viene arrotondato solamente una volta, usando il modo di arrotondamento corrispondente a `FLT_ROUNDS`. Può verificarsi un errore di arrotondamento.

23.4

fmax *Massimo in virgola mobile (C99)*

<math.h>

```
double fmax(double x, double y);
```

fmaxf

```
float fmaxf(float x, float y);
```

fmaxl

```
long double fmaxl(long double x, long double y);
```

Restituisce

Il massimo tra x e y . Se uno degli argomenti è un NaN e l'altro è numerico, viene restituito il valore numerico.

23.4

fmin *Minimo in virgola mobile (C99)*

<math.h>

```
double fmin(double x, double y);
```

fminf

```
float fminf(float x, float y);
```

fminl

```
long double fminl(long double x, long double y);
```

Restituisce

Il minimo tra x e y . Se uno degli argomenti è un NaN e l'altro è numerico, viene restituito il valore numerico.

23.4

fmod	<i>Modulo in virgola mobile</i>	<math.h>
<code>double fmod(double x, double y);</code>		
		fmodf
<code>float fmodf(float x, float y);</code>		
		fmodl
<code>long double fmodl(long double x, long double y);</code>		
<i>Restituisce</i>		
Il resto ottenuto quando <i>x</i> viene diviso per <i>y</i> . Se <i>y</i> è uguale a zero, si verifica un errore di dominio oppure viene restituito il valore zero.		
		23.3

fopen	<i>Apertura di un file</i>	<stdio.h>
<code>FILE *fopen(const char * restrict filename, const char * restrict mode);</code>		
Apre il file il cui nome è puntato da <i>filename</i> e lo associa a uno stream. L'argomento <i>mode</i> specifica la modalità di apertura nella quale deve essere aperto il file. Azzerà gli indicatori di errore e di fine file associati dallo stream.		
<i>Restituisce</i>		
Un puntatore a un file che può essere usato quando vengono eseguite delle successive operazioni sul file stesso. Se il file non può essere aperto restituisce un puntatore nullo.		
		22.2

fpclassify	<i>Classificazione floating point (C99)</i>	<math.h>
<code>int fpclassify(real-floating x);</code>		
	<i>macro</i>	
<i>Restituisce</i>		
Il valore FP_INFINITE, FP_NAN, FP_NORMAL, FP_SUBNORMAL o FP_ZERO, a seconda che <i>x</i> sia rispettivamente infinito, NaN, normale, subnormale o zero.		
		23.4

fprintf	<i>Scrittura formattata su file</i>	<stdio.h>
<code>int fprintf(FILE * restrict stream, const char * restrict format, ...);</code>		
Scrive l'output nello stream puntato da <i>stream</i> . La stringa puntata da <i>format</i> specifica come debbano essere visualizzati gli argomenti successivi.		
<i>Restituisce</i>		
Il numero di caratteri scritti. Se si verifica un errore restituisce un valore negativo.		
		22.3

fputc	<i>Scrittura di un carattere su file</i>	<stdio.h>
<code>int fputc(int c, FILE *stream);</code>		
Scrive il carattere <i>c</i> nello stream puntato da <i>stream</i> .		
<i>Restituisce</i>		
Il carattere scritto <i>c</i> . Se si verifica un errore nella scrittura, la funzione imposta l'indicatore di errore associato allo stream e restituisce EOF.		
		22.4

fputs	<i>Scrittura di una stringa su file</i>	<stdio.h>
<code>int fputs(const char * restrict s, FILE * restrict stream);</code>		
Scrive la stringa puntata da <i>s</i> nello stream puntato da <i>stream</i> .		
<i>Restituisce</i>		
Un valore non negativo se la chiamata ha successo. Se si verifica un errore nella scrittura restituisce il valore EOF.		
		22.5

fputwc	<i>Scrittura di un wide character su file (C99)</i>	<wchar.h>
<code>wint_t fputwc(wchar_t c, FILE *stream);</code>		
La versione wide character della <i>fputc</i> .		
		25.5

fputws	<i>Scrittura una stringa wide su file (C99)</i>	<wchar.h>
<code>int fputws(const wchar_t * restrict s, FILE * restrict stream);</code>		
La versione wide character della <i>fputs</i> .		
		25.5

fread *Lettura di un blocco da file*

<stdio.h>

```
size_t fread(void * restrict ptr, size_t size,
             size_t nmemb, FILE * restrict stream);
```

Cerca di leggere `nmemb` elementi, ognuno di lunghezza pari a `size` byte, dallo stream puntato dalla `stream` e salvarli nel vettore puntato da `ptr`.

Restituisce

Il numero di elementi letti. Questo numero sarà minore di `nmemb` se la funzione incontra la fine del file o legge l'errore di lettura. Restituisce lo zero se `nmemb` o `size` sono uguali a zero. 22.6

free *Liberare un blocco di memoria*

<stdlib.h>

```
void free(void *ptr);
```

Rilascia il blocco di memoria puntato da `ptr` (se `ptr` è un puntatore nullo la chiamata non ha effetto). Il blocco deve essere stato allocato da una chiamata alle funzioni `calloc`, `malloc` o `realloc`. 17.4

freopen *Riapertura di un file*

<stdio.h>

```
FILE *freopen(const char * restrict filename,
              const char * restrict mode,
              FILE * restrict stream);
```

Chiude il file associato a `stream`, successivamente apre il file il cui nome è puntato da `filename` e lo associa a `stream`. Il parametro `mode` ha lo stesso significato di una chiamata alla `fopen`. *Modifica del C99*: se `filename` è un puntatore nullo, la funzione cerca di modificare la modalità dello stream facendola diventare come quella specificata da `mode`.

Restituisce

Il valore di `stream` se l'operazione ha successo. Restituisce un puntatore nullo se il file non può essere aperto. 22.2

frexp *Divisione in mantissa ed esponente*

<math.h>

```
double frexp(double value, int *exp);
```

```
float frexpf(float value, int *exp);
```

```
long double frexpl(long double value, int *exp);
```

frexpf**frexpl**

Divide `value` in una mantissa `f` e in un esponente `n` in modo che

$$\text{value} = f \times 2^n$$

`f` viene normalizzata in modo che $0.5 \leq f < 1$ o $f = 0$. Salva `n` nell'oggetto puntato da `exp`.

Restituisce

`f`, la mantissa del valore `value`. 23.3

fscanf *Lettura formattata da file*

<stdio.h>

```
int fscanf(FILE * restrict stream,
           const char * restrict format, ...);
```

Legge gli oggetti di input dallo stream puntato da `stream`. La stringa puntata da `format` specifica il formato degli oggetti che devono essere letti. Gli argomenti che seguono `format` puntano agli oggetti nei quali devono essere salvati gli oggetti letti.

Restituisce

Il numero degli oggetti di input letti e salvati con successo. Restituisce il valore EOF se si verifica qualche problema nell'input prima che uno qualsiasi degli oggetti possa essere letto. 22.3

fseek *File Seek*

<stdio.h>

```
int fseek(FILE *stream, long int offset, int whence);
```

Modifica la posizione dell'indicatore associato allo stream puntato da `stream`. Se `whence` è uguale a `SEEK_SET`, la nuova posizione è l'inizio del file più `offset` byte. Se `whence` è uguale a `SEEK_CUR`, la nuova posizione è la posizione corrente più `offset` byte. Se `whence` è uguale a `SEEK_END`, la nuova posizione è la fine del file più `offset` byte. Il valore di `offset` può essere negativo. Per gli stream di testo, `offset` deve essere uguale a zero oppure `whence` deve essere uguale a `SEEK_SET` e `offset` deve essere un valore ottenuto da una previa chiamata alla

`ftell`. Per degli stream binari, la `fseek` potrebbe non supportare delle chiamate nella quali `whence` è uguale a `SEEK_END`.

Restituisce

Il valore zero se l'operazione ha avuto successo, altrimenti restituisce un valore diverso da zero. 22.7

fsetpos *Impostare la posizione del file* <stdio.h>

```
int fsetpos(FILE *stream, const fpos_t *pos);
```

Imposta l'indicatore di posizione del file per lo stream puntato da `stream` in accordo con il valore puntato da `pos` (ottenuto da una precedente chiamata alla `fgetpos`).

Restituisce

Il valore zero se la chiamata ha successo. Se la chiamata fallisce, la funzione restituisce un valore diverso da zero e salva nella variabile `errno` un valore positivo definito dall'implementazione. 22.7

ftell *Determinazione della posizione del file* <stdio.h>

```
long int ftell(FILE *stream);
```

Restituisce

L'indicatore della posizione del file per lo stream puntato da `stream`. Se la chiamata fallisce, la funzione restituisce `-1L` e salva nella variabile `errno` un valore positivo definito dall'implementazione. 22.7

fwide *Ottenere e impostare l'orientamento del file (C99)* <wchar.h>

```
int fwide(FILE *stream, int mode);
```

Determina l'orientamento corrente di uno stream e, se lo si desidera, cerca di impostare il suo orientamento. Se `mode` è maggiore di zero, `fwide` cerca di rendere lo stream wide-oriented nel caso questo non avesse un orientamento. Se `mode` è minore di zero, cerca di rendere lo stream byte-oriented nel caso questo non avesse un orientamento. Se `mode` è uguale a zero, l'orientamento non viene modificato.

Restituisce

Un valore positivo se dopo la chiamata lo stream è wide-oriented, un valore negativo se è byte-oriented e pari a zero se non ha alcun orientamento. 25.5

fwprintf *Scrittura formattata di wide character (C99)* <wchar.h>

```
int fwprintf(FILE * restrict stream,
             const wchar_t * restrict format, ...);
```

La versione wide character della `fprintf`. 25.5

fwrite *Scrittura di un blocco su file* <stdio.h>

```
size_t fwrite(const void * restrict ptr, size_t size,
             size_t nmemb, FILE * restrict stream);
```

Scrive `nmemb` elementi, ognuno lungo `size` byte, dal vettore puntato da `ptr` allo stream puntato da `stream`.

Restituisce

Il numero di elementi effettivamente scritti. Questo numero sarà minore di `nmemb` se si verifica un errore nella scrittura. Nel C99 restituisce il valore zero se `nmemb` o `size` sono uguali a zero. 22.6

fwscanf *Lettura formattata di wide character da file (C99)* <wchar.h>

```
int fwscanf(FILE * restrict stream,
            const wchar_t * restrict format, ...);
```

Versione wide character della `fscanf`. 25.5

getc *Lettura di un carattere da file* <stdio.h>

```
int getc(FILE *stream);
```

Legge un carattere dallo stream puntato da `stream`. *Nota:* normalmente la `getc` viene implementata come una macro, può valutare `stream` più di una volta.

Restituisce

Il carattere letto da uno stream. Se la `getc` incontra la fine dello stream, imposta l'indicatore di end-of-file e restituisce il valore EOF. Se si verifica un errore di lettura, la `getc` imposta l'indicatore di errore dello stream e restituisce il valore EOF. 22.4

getchar *Lettura di un carattere* <stdio.h>

```
int getchar(void);
```

Legge un carattere dallo stream `stdin`. *Nota:* normalmente la `getchar` viene implementata come una macro.

Restituisce

I caratteri letti dallo stream. Se la funzione incontra la fine dello stream, imposta l'indicatore di end-of-file e restituisce il valore EOF. Se si verifica un errore di lettura, la `getchar` imposta l'indicatore di errore e restituisce il valore EOF.

7.3, 22.4

getenv *Ottenere la stringa d'ambiente*

<stdlib.h>

```
char *getenv(const char *name);
```

Cerca all'interno dell'elenco d'ambiente del sistema operativo per vedere se qualche stringa corrisponde con quella puntata da `name`.

Restituisce

Un puntatore alla stringa associata con il nome corrispondente. Se non viene trovata nessuna corrispondenza restituisce un puntatore nullo.

26.2

gets *Lettura di una stringa*

<stdio.h>

```
char *gets(char *s);
```

Legge dei caratteri dallo stream `stdin` e li salva nel vettore puntato da `s`. La lettura si interrompe al primo carattere new-line (che viene scartato) o alla fine del file. La funzione aggiunge il carattere null alla fine della stringa.

Restituisce

`s`, ovvero un puntatore al vettore nel quale viene salvato l'input. Restituisce un puntatore nullo nel caso si verificasse un errore nella lettura o venisse incontrata la fine dello stream prima di aver salvato qualsiasi carattere.

13.3, 22.5

getwc *Lettura di un wide character da file (C99)*

<wchar.h>

```
wint_t getwc(FILE *stream);
```

Versione wide character della funzione `getc`.

25.5

getwchar *Lettura di un wide character (C99)*

<wchar.h>

```
wint_t getwchar(void);
```

Versione wide character della funzione `getchar`.

25.5

gmtime *Converte un'ora di tipo calendar time in una di tipo broken-down UTC*

<time.h>

```
struct tm *gmtime(const time_t *timer);
```

Restituisce

Un puntatore a una struttura contenente un'ora di tipo broken-down UTC equivalente all'ora di tipo calendar time puntata da `timer`. Se l'ora calendar time non può essere convertito a UTC restituisce un puntatore nullo.

26.3

hypot *Ipotenusa (C99)*

<math.h>

```
double hypot(double x, double y);
```

hypotf

```
float hypotf(float x, float y);
```

hypotl

```
long double hypotl(long double x, long double y);
```

Restituisce

$\sqrt{x^2 + y^2}$ (l'ipotenusa del triangolo retto con lati `x` e `y`). Può verificarsi un errore di intervallo.

23.4

ilogb *Unbiased Exponent (C99)*

<math.h>

```
int ilogb(double x);
```

ilogbf

```
int ilogbf(float x);
```

ilogbl

```
int ilogbl(long double x);
```

Restituisce

L'esponente di `x` inteso come intero con segno, è equivalente a chiamare la corrispondente funzione `logb` e fare il casting al tipo `int` del valore restituito. Restituisce `FP_ILOGB0` se `x` è uguale a zero, `INT_MAX` se `x` è infinito e `FP_ILOGBNAN` se `x` è un NaN. In questi casi può verificarsi un errore di dominio o di intervallo.

23.4

imaxabs *Valore assoluto per gli interi con dimensione più grande (C99)*

<inttypes.h>

```
intmax_t imaxabs(intmax_t j);
```

Restituisce

Il valore assoluto di j . Il comportamento non è definito se il valore assoluto di j non può essere rappresentato. 27.2

imaxdiv *Divisione per gli interi di dimensione più grande (C99)* <inttypes.h>

`imaxdiv_t imaxdiv(intmax_t numer, intmax_t denom);`

Restituisce

Una struttura di tipo `imaxdiv_t` contenente dei membri chiamati `quot` (il quoziente ottenuto dividendo `numer` per `denom`) e `rem` (il resto). Il comportamento non è definito nel caso una delle due parti del risultato non possa essere rappresentata. 27.2

isalnum *Test per carattere alfanumerico* <ctype.h>

`int isalnum(int c);`

Restituisce

Un valore diverso da zero se c è un carattere alfanumerico e il valore zero altrimenti (c è alfanumerico se `isalpha(c)` è vero o lo è `isdigit(c)`). 23.5

isalpha *Test per carattere alfabetico* <ctype.h>

`int isalpha(int c);`

Restituisce

Un valore diverso da zero se c è un carattere alfabetico e il valore zero altrimenti. Nella localizzazione "C", il carattere c è alfabetico se `islower(c)` è vero oppure lo è `isupper(c)`. 23.5

isblank *Test per carattere di spazio (C99)* <ctype.h>

`int isblank(int c);`

Restituisce

Un valore diverso da zero se c è un carattere corrispondente a uno spazio che viene usato per separate le parole all'interno di una riga di testo. Nella localizzazione "C" vengono considerati caratteri di spazio: lo spazio vero e proprio (' ') e la tabulazione orizzontale ('\t'). 23.5

iscntrl *Test per carattere di controllo* <ctype.h>

`int iscntrl(int c);`

Restituisce

Un valore diverso da zero se c è un carattere di controllo, altrimenti restituisce il valore zero. 23.5

isdigit *Test per carattere corrispondente a una cifra* <ctype.h>

`int isdigit(int c);`

Restituisce

Un valore diverso da zero se c è una cifra decimale, altrimenti restituisce il valore zero. 23.5

isfinite *Test per numero finito (C99)* <math.h>

`int isfinite(real-floating x);` *macro*

Restituisce

Un valore diverso da zero se x è un numero finito (zero, subnormale o normale, ma non infinito o NaN) e zero altrimenti. 23.4

isgraph *Test per carattere grafico* <ctype.h>

`int isgraph(int c);`

Restituisce

Un valore diverso da zero se c è un carattere stampabile (ad eccezione dello spazio) e pari a zero altrimenti. 23.5

isgreater *Test per maggiore di (C99)* <math.h>

`int isgreater(real-floating x, real-floating y);` *macro*

Restituisce

$(x) > (y)$. A differenza dell'operatore $>$, non solleva l'eccezione floating point *invalid* nel caso in cui uno o entrambi i suoi argomenti siano NaN. 23.4

isgreaterequal *Test per maggiore o uguale a (C99)* <math.h>

`int isgreaterequal(real-floating x, real-floating y);` *macro*

Restituisce

$(x) \geq (y)$. A differenza dell'operatore \geq , non solleva l'eccezione floating point *invalid* nel caso in cui uno o

entrambi i suoi argomenti siano NaN.		23.4
isinf Test per infinito (C99)	<math.h>	
<code>int isinf(real-floating x);</code>	macro	
<i>Restituisce</i>		
Un valore diverso da zero se x è infinito (positivo o negativo) e pari a zero altrimenti.		23.4
isless Test per minore di (C99)	<math.h>	
<code>int isless(real-floating x, real-floating y);</code>	macro	
<i>Restituisce</i>		
$(x) < (y)$. A differenza dell'operatore $<$, non solleva l'eccezione floating point <i>invalid</i> nel caso in cui uno o entrambi i suoi argomenti siano NaN.		23.4
islessequal Test per minore o uguale a (C99)	<math.h>	
<code>int islessequal(real-floating x, real-floating y);</code>	macro	
<i>Restituisce</i>		
$(x) \leq (y)$. A differenza dell'operatore \leq , <i>islessequal</i> non solleva l'eccezione floating point <i>invalid</i> nel caso in cui uno o entrambi i suoi argomenti siano NaN.		23.4
islessgreater Test per minore di o maggiore a (C99)	<math.h>	
<code>int islessgreater(real-floating x, real-floating y);</code>	macro	
<i>Restituisce</i>		
$(x) < (y) \parallel (x) > (y)$. A differenza di questa espressione, non solleva l'eccezione floating point <i>invalid</i> nel caso in cui uno o entrambi i suoi argomenti siano NaN, inoltre x e y vengono calcolati solamente una volta.		23.4
islower Test per lettera minuscola	<ctype.h>	
<code>int islower(int c);</code>		
<i>Restituisce</i>		
Un valore diverso da zero se c è una lettera minuscola e pari a zero altrimenti.		23.5
isnan Test per NaN (C99)	<math.h>	
<code>int isnan(real-floating x);</code>	macro	
<i>Restituisce</i>		
Un valore diverso da zero se x è un valore NaN value e pari a zero altrimenti.		23.4
isnormal Test per un numero normale (C99)	<math.h>	
<code>int isnormal(real-floating x);</code>	macro	
<i>Restituisce</i>		
Un valore diverso da zero se x possiede un valore normale (ovvero che non sia: zero, subnormale, infinito o NaN), altrimenti restituisce il valore zero.		23.4
isprint Test per carattere stampabile	<ctype.h>	
<code>int isprint(int c);</code>		
<i>Restituisce</i>		
Un valore diverso da zero se c è un carattere stampabile (incluso lo spazio) e pari a zero altrimenti.		23.5
ispunct Test per carattere di punteggiatura	<ctype.h>	
<code>int ispunct(int c);</code>		
<i>Restituisce</i>		
Un valore diverso da zero se c è un carattere di punteggiatura e pari a zero altrimenti. Tutti i caratteri stampabili ad eccezione dello spazio (' ') e dei caratteri alfanumerici sono considerati punteggiatura. <i>Modifiche del C99</i> : nella localizzazione "C", sono considerati caratteri di punteggiatura tutti i caratteri stampabili ad eccezione di quelli per i quali <i>isspace</i> è vera o è vera <i>isalnum</i> .		23.5
isspace Test per carattere di spazio bianco	<ctype.h>	
<code>int isspace(int c);</code>		
<i>Restituisce</i>		
Un valore diverso da zero se c è un carattere di spazio bianco e pari a zero altrimenti. Nella localizzazione "C" i caratteri considerati di spazio bianco sono: lo spazio (' '), form feed ('\f'), new-line ('\n'), carriage return ('\r'),		

la tabulazione orizzontale ('\t') e la tabulazione verticale ('\v').	23.5
isunordered <i>Test per non ordinati (C99)</i>	<math.h>
<i>int isunordered(real-floating x, real-floating y);</i>	<i>macro</i>
<i>Restituisce</i>	
Il valore 1 se x e y sono non ordinati (almeno uno è NaN) e 0 altrimenti.	23.4
isupper <i>Test per lettera maiuscola</i>	<ctype.h>
<i>int isupper(int c);</i>	
<i>Restituisce</i>	
Un valore diverso da zero se c è una lettera maiuscola e pari a zero altrimenti.	23.5
iswalnum <i>Test per wide character alfanumerico (C99)</i>	<wctype.h>
<i>int iswalnum(wint_t wc);</i>	
<i>Restituisce</i>	
Un valore diverso da zero se wc è alfanumerico, altrimenti restituisce il valore zero (wc è alfanumerico se iswalpna(wc) è vero o se è vero iswdigit(wc)).	25.6
iswalpha <i>Test per wide character alfabetico (C99)</i>	<wctype.h>
<i>int iswalpha(wint_t wc);</i>	
<i>Restituisce</i>	
Un valore diverso da zero se wc è un carattere alfabetico, altrimenti restituisce il valore zero (wc è alfabetico se iswupper(wc) o iswlower(wc) sono veri, oppure se wc è uno degli elementi di un insieme di wide character alfabetici specifici per una localizzazione per il quale nessuna tra le funzioni iswcntrl, iswdigit, iswpunct e iswspace è vera).	25.6
iswblank <i>Test per wide character corrispondenti a uno spazio (C99)</i>	<wctype.h>
<i>int iswblank(wint_t wc);</i>	
<i>Restituisce</i>	
Un valore diverso da zero se wc è un wide character standard di spazio o se è uno degli elementi di un insieme di wide character specifici per una localizzazione per il quale la funzione iswspace è vera e che viene usato per separare le parole all'intero di una riga di testo. Nella localizzazione "C", iswblank restituisce il valore true solo per i caratteri standard di spazio: lo spazio (L' ') e la tabulazione orizzontale (L'\t').	25.6
iswcntrl <i>Test per wide character di controllo (C99)</i>	<wctype.h>
<i>int iswcntrl(wint_t wc);</i>	
<i>Restituisce</i>	
Un valore diverso da zero se wc è un wide character di controllo, altrimenti restituisce il valore zero.	25.6
iswctype <i>Controllo del tipo di wide character(C99)</i>	<wctype.h>
<i>int iswctype(wint_t wc, wctype_t desc);</i>	
<i>Restituisce</i>	
Un valore diverso da zero se il wide character wc ha la proprietà descritta da desc (desc deve essere un valore restituito da una chiamata alla wctype; l'impostazione corrente per la categoria LC_CTYPE deve essere la stessa durante entrambe le chiamate). Altrimenti restituisce il valore zero.	25.6
iswdigit <i>Test per cifra wide character (C99)</i>	<wctype.h>
<i>int iswdigit(wint_t wc);</i>	
<i>Restituisce</i>	
Un valore diverso da zero se wc corrisponde a una cifra decimale e pari a zero altrimenti.	25.6
iswgraph <i>Test per wide character grafico (C99)</i>	<wctype.h>
<i>int iswgraph(wint_t wc);</i>	
<i>Restituisce</i>	
Un valore diverso da zero se iswprint(wc) è vero e se iswspace(wc) è falso. Negli altri casi restituisce il valore zero.	25.6
iswlower <i>Test per wide character minuscolo (C99)</i>	<wctype.h>
<i>int iswlower(wint_t wc);</i>	
<i>Restituisce</i>	

Un valore diverso da zero se `wc` corrisponde a una lettera minuscola oppure se è uno degli elementi di un insieme di wide character specifici per una particolare localizzazione per il quale una delle funzioni `iswcntrl`, `iswdigit`, `iswpunct` e `iswspace` è vera. Negli altri casi restituisce il valore zero. 25.6

`iswprint` *Test per wide character stampabile (C99)* <wctype.h>

`int iswprint(wint_t wc);`

Restituisce

Un valore diverso da zero se `wc` è un wide character stampabile e zero altrimenti. 25.6

`iswpunct` *Test per wide character di punteggiatura (C99)* <wctype.h>

`int iswpunct(wint_t wc);`

Restituisce

Un valore diverso da zero se `wc` è un wide character stampabile che appartiene a un insieme di wide character di punteggiatura specifici per una certa localizzazione per il quale `iswspace` e `iswalnum` hanno valore `true`. Negli altri casi restituisce il valore zero. 25.6

`iswspace` *Test per wide character di spazio bianco (C99)* <wctype.h>

`int iswspace(wint_t wc);`

Restituisce

Un valore diverso da zero se `wc` è uno degli elementi di un insieme di wide character di spazio specifici per una localizzazione per il quale nessuna delle funzioni `iswalnum`, `iswgraph` e `iswpunct` è vera. Negli altri casi restituisce il valore zero. 25.6

`iswupper` *Test per wide character maiuscolo (C99)* <wctype.h>

`int iswupper(wint_t wc);`

Restituisce

Un valore diverso da zero se `wc` corrisponde a una lettera maiuscola o è uno degli elementi di un insieme di wide character specifici per una localizzazione per il quale nessuna delle funzioni `iswcntrl`, `iswdigit`, `iswpunct` e `iswspace` è vera. Negli altri casi restituisce il valore zero. 25.6

`iswxdigit` *Test per wide character corrispondente a una cifra esadecimale (C99)* <wctype.h>

`int iswxdigit(wint_t wc);`

Restituisce

Un valore diverso da zero se `wc` corrisponde a una cifra esadecimale (0-9, a-f, A-F), altrimenti restituisce il valore zero. 25.6

`isxdigit` *Test per cifra esadecimale* <ctype.h>

`int isxdigit(int c);`

Restituisce

Un valore diverso da zero se `c` è una cifra esadecimale (0-9, a-f, A-F) e zero altrimenti. 23.5

`labs` *Valore assoluto long int* <stdlib.h>

`long int labs(long int j);`

Restituisce

Valore assoluto di `j`. Il comportamento non è definito se il valore assoluto di `j` non può essere rappresentato. 26.2

`ldexp` *Combinare esponente e mantissa* <math.h>

`double ldexp(double x, int exp);`

`ldexpf`

`float ldexpf(float x, int exp);`

`ldexpl`

`long double ldexpl(long double x, int exp);`

Restituisce

$x \times 2^{\text{exp}}$. Può verificarsi un errore di intervallo. 23.3

`ldiv` *Divisione long int* <stdlib.h>

`ldiv_t ldiv(long int numer, long int denom);`

Restituisce

Una struttura `ldiv_t` contenente dei membri chiamati `quot` (il quoziente ottenuto dividendo `numer` per `denom`) e `rem` (il resto). Il comportamento non è definito se una delle due componenti del risultato non può essere rappresentata.

lgamma *Logaritmo della funzione gamma (C99)*

<math.h>

```
double lgamma(double x);
```

lgammaf

```
float lgammaf(float x);
```

lgammal

```
long double lgammal(long double x);
```

Restituisce

$\ln(\Gamma(x))$, dove Γ è la funzione gamma. Un errore di intervallo si verifica se x è troppo grande e mentre può verificarsi se x è un intero negativo o uguale a zero.

23.4

llabs *Valore assoluto long long int (C99)*

<stdlib.h>

```
long long int llabs(long long int j);
```

Restituisce

Il valore assoluto di j . Il comportamento non è definito se il valore assoluto di j non può essere rappresentato.

26.2

lldiv *Divisione intera long long int (C99)*

<stdlib.h>

```
lldiv_t lldiv(long long int numer,
              long long int denom);
```

Restituisce

Una struttura `lldiv_t` contenente dei membri chiamati `quot` (il quoziente ottenuto quando `numer` viene diviso per `denom`) e `rem` (il resto). Il comportamento non è definito se una delle due componenti del risultato non può essere rappresentata.

26.2

llrint *Arrotondamento a long long int usando la direzione corrente (C99)*<math.h>

```
long long int llrint(double x);
```

llrintf

```
long long int llrintf(float x);
```

llrintl

```
long long int llrintl(long double x);
```

Restituisce

x arrotondato al più vicino intero usando la corrente direzione di arrotondamento. Se il valore arrotondato è al di fuori dell'intervallo rappresentabile dal tipo `long long int`, il risultato non è specificato e può verificarsi un errore di dominio o un errore di intervallo.

23.4

llround *Arrotondamento al più vicino intero long long int (C99)*

<math.h>

```
long long int llround(double x);
```

llroundf

```
long long int llroundf(float x);
```

llroundl

```
long long int llroundl(long double x);
```

Restituisce

x arrotondato al più vicino intero, con i casi a metà strada arrotondati distanziandosi dallo zero. Se il valore arrotondato si trova al di fuori dell'intervallo del tipo `long long int`, il risultato non è specificato e può verificarsi un errore di dominio o di intervallo.

23.4

localeconv *Ottenere le convenzioni locali*

<locale.h>

```
struct lconv *localeconv(void);
```

Restituisce

Un puntatore a una struttura contenente le informazioni sulla localizzazione corrente.

25.1

localtime *Conversione di un ora di tipo calendar time in un'ora locale di tipo broken down* <time.h>

```
struct tm *localtime(const time_t *timer);
```

Restituisce

Un puntatore a una struttura contenente un'ora locale del formato broken down che è equivalente all'ora di tipo `calendar time` puntata da `timer`. Se l'ora `calendar time` non può essere convertito in un'ora locale restituisce un puntatore nullo.

26.3

log <i>Logaritmo naturale</i>	<math.h>
double log(double x);	logf
float logf(float x);	logl
long double logl(long double x);	
<i>Restituisce</i>	
Il logaritmo con base e di x . Se x è negativo si verifica un errore di dominio. Se x è uguale a zero può verificarsi un errore di intervallo.	23.3

log10 <i>Logaritmo comune</i>	<math.h>
double log10(double x);	log10f
float log10f(float x);	log10l
long double log10l(long double x);	
<i>Restituisce</i>	
Il logaritmo in base 10 di x . Se x è negativo si verifica un errore di dominio. Se x è uguale a zero può verificarsi un errore di intervallo.	23.3

log1p <i>Logaritmo naturale dell'argomento più uno (C99)</i>	<math.h>
double log1p(double x);	log1pf
float log1pf(float x);	log1pl
long double log1pl(long double x);	
<i>Restituisce</i>	
Il logaritmo in base e di $1 + x$. Se x è minore di -1 si verifica un errore di dominio. Se x è uguale a -1 può verificarsi un errore di intervallo.	23.4

log2 <i>Logaritmo base 2 (C99)</i>	<math.h>
double log2(double x);	log2f
float log2f(float x);	log2l
long double log2l(long double x);	
<i>Restituisce</i>	
Il logaritmo in base 2 di x . Se x è negativo si verifica un errore di dominio. Se x è uguale a zero può verificarsi un errore di intervallo.	23.4

logb <i>Esponente indipendente dalla radice (C99)</i>	<math.h>
double logb(double x);	logbf
float logbf(float x);	logbl
long double logbl(long double x);	
<i>Restituisce</i>	
$\log_2(x)$, dove r è la radice dell'aritmetica in virgola mobile (definite dalla macro FLT_RADIX che tipicamente ha valore 2). Se x è uguale a zero può verificarsi un errore di dominio o un errore di intervallo.	23.4

longjmp <i>Salto non locale</i>	<setjmp.h>
void longjmp(jmp_buf env, int val);	
Ripristina l'ambiente memorizzato all'interno di env e ritorna dalla chiamata alla setjmp che originariamente ha salvato l'ambiente nello stesso env. Se val è diverso da zero, questo sarà il valore restituito dalla setjmp, altrimenti se val è uguale a 0, la setjmp restituisce 1.	24.4

lrint <i>Arrotonda a un long int usando la direzione corrente (C99)</i>	<math.h>
long int lrint(double x);	

lrintf`long int lrintf(float x);`**lrintl**`long int lrintl(long double x);`**Restituisce**

x arrotondato all'intero più vicino usando la corrente direzione di arrotondamento. Se il valore arrotondato è al di fuori dell'intervallo rappresentabile con il tipo `long int`, il risultato non è specificato e può verificarsi un errore di dominio o un errore di intervallo. 23.4

lround Arrotondamento al più vicino long int (C99)

<math.h>

`long int lround(double x);`**lroundf**`long int lroundf(float x);`**lroundl**`long int lroundl(long double x);`**Restituisce**

x arrotondato all'intero più vicino, con i casi a metà strada arrotondati allontanandosi dallo zero. Se il valore arrotondato è al di fuori dell'intervallo rappresentabile con il tipo `long int`, il risultato non è specificato e può verificarsi un errore di dominio o un errore di intervallo. 23.4

malloc Allocazione di un blocco di memoria

<stdlib.h>

`void *malloc(size_t size);`

Alloca un blocco di memoria con `size` byte. Il blocco non viene azzerato.

Restituisce

Un puntatore all'inizio del blocco. Se non può essere allocato un blocco della dimensione richiesta la funzione restituisce un puntatore nullo. 17.2

mblen Lunghezza di un carattere multibyte

<stdlib.h>

`int mblen(const char *s, size_t n);`**Restituisce**

Se `s` è un puntatore nullo restituisce un valore diverso da zero o uguale a zero a seconda che i caratteri multibyte abbiano o meno delle codifiche dipendenti dallo stato. Se `s` punta a un carattere null, la funzione restituisce il valore zero. Negli altri casi la funzione restituisce il numero di byte appartenenti al carattere multibyte puntato da `s`. Restituisce -1 se i successivi `n` o meno byte non formano un carattere multibyte valido. 25.2

mbrlen Lunghezza di un carattere multibyte – Riavviabile (C99)

<wchar.h>

```
size_t mbrlen(const char * restrict s, size_t n,
              mbstate_t * restrict ps);
```

Determina il numero di byte nel vettore puntato da `s` che sono necessari per completare un carattere multibyte. L'argomento `ps` deve puntare a un oggetto di tipo `mbstate_t` contenente lo stato di conversione corrente. Una chiamata alla `mbrlen` è equivalente a

`mbrtowc(NULL, s, n, ps)`

ad eccezione del fatto che se `ps` è un puntatore nullo, viene usato al suo posto l'indirizzo di un oggetto interno.

RestituisceVedi `mbrtowc`.25.5**mbrtowc** Conversione da carattere multibyte a wide character – Riavviabile <wchar.h> (C99)

```
size_t mbrtowc(wchar_t * restrict pwc,
               const char * restrict s, size_t n,
               mbstate_t * restrict ps);
```

Se `s` è un puntatore nullo, una chiamata a questa funzione è equivalente a

`mbrtowc(NULL, "", 1, ps)`

Negli altri casi la `mbrtowc` esamina nel vettore puntato da `s` fino a `n` byte per vedere se questi completano un carattere multibyte. Se è così il carattere multibyte viene convertito in un wide character. Se `pwc` non è un puntatore nullo, il

wide character viene salvato nell'oggetto puntato dallo stesso `pwc`. Il valore di `ps` deve essere un puntatore a un oggetto di tipo `mbstate_t` contenente lo stato di conversione corrente. Se `ps` è un puntatore nullo, la funzione utilizza un oggetto interno per salvare il corrente stato di conversione. Se il risultato della conversione è un wide character null, l'oggetto `mbstate_t` utilizzato durante la chiamata viene lasciato allo stato di conversione iniziale.

Restituisce

0 se la conversione produce un wide character null. Restituisce un numero compreso tra 1 ed `n` se la conversione produce un wide character diverso da null, dove il valore restituito è il numero di byte usati per completare il carattere multibyte. Restituisce `(size_t) (-2)` se gli `n` byte puntato da `s` non sono stati sufficienti a completare un carattere multibyte. Se si verifica un errore di codifica restituisce `(size_t) (-1)` e salva il valore `EILSEQ` nella variabile `errno`.

25.5

mbsinit *Test per lo stato di conversione iniziale (C99)*

<wchar.h>

```
int mbsinit(const mbstate_t *ps);
```

Restituisce

Un valore diverso da zero se `ps` è un puntatore nullo o se punta a un oggetto `mbstate_t` che descrive uno stato di conversione iniziale. Negli altri casi restituisce il valore zero.

25.5

mbsrtowcs *Conversione da stringa multibyte a stringa wide – Riavviabile (C99)*<wchar.h>

```
size_t mbsrtowcs(wchar_t * restrict dst,
                 const char ** restrict src,
                 size_t len, mbstate_t * restrict ps);
```

Converte una sequenza di caratteri multibyte a partire dal vettore indirettamente puntato da `src` in una sequenza di wide character corrispondenti. Il parametro `ps` deve puntare a un oggetto di tipo `mbstate_t` contenente lo stato di conversione corrente. Se l'argomento corrispondente a `ps` è un puntatore nullo, la funzione utilizza un oggetto interno per salvare lo stato di conversione. Se `dst` non è un puntatore nullo, i caratteri convertiti vengono salvati nel vettore a cui questo punta. La conversione continua fino al carattere null di termine, il quale viene salvato. La conversione si ferma prima se viene incontrata una sequenza di byte che non forma un valido carattere multibyte o, nel caso `dst` non fosse un puntatore nullo, quando nel vettore sono stati salvati `len` wide character. Se `dst` non è un puntatore nullo, all'oggetto puntato da `src` viene assegnato o un puntatore nullo (se è stato raggiunto il carattere null di termine) o l'indirizzo immediatamente successivo all'ultimo carattere multibyte convertito (se presente). Se la conversione finisce a un carattere null e se `dst` non è un puntatore nullo, lo stato risultate della conversione è quello iniziale.

Restituisce

Il numero di caratteri multibyte che sono stati convertiti con successo, non includendo nessun carattere null di termine. Se viene incontrato un carattere multibyte non valido restituisce `(size_t) (-1)` e salva `EILSEQ` in `errno`.

25.5

mbstowcs *Conversione da stringa multibyte a stringa wide*

<stdlib.h>

```
size_t mbstowcs(wchar_t * restrict pwcs,
                const char * restrict s, size_t n);
```

Converte la sequenza di caratteri multibyte puntati da `s` in una sequenza di wide character salvando al massimo `n` wide character all'interno del vettore puntato da `pwcs`. La conversione ha termine se viene incontrato il carattere null, il quale viene convertito in un wide character null.

Restituisce

Il numero degli elementi del vettore modificati non includendo nel conto il wide character null (se presente). Se viene incontrato un carattere multibyte non valido restituisce `(size_t) (-1)`.

25.2

mbtowc *Conversione di un carattere multibyte in un wide character* <stdlib.h>

```
int mbtowc(wchar_t * restrict pwc,
            const char * restrict s, size_t n);
```

Se `s` non è un puntatore nullo converte il carattere multibyte puntato da `s` in un wide character. Verranno esaminati un massimo di `n` byte. Se il carattere multibyte è valido e `pwc` non è un puntatore nullo, la funzione salva il valore del wide character nell'oggetto puntato da `pwc`.

Restituisce

Se `s` è un puntatore nullo restituisce un valore diverso da zero o uguale a zero a seconda che i caratteri multibyte abbiano o meno delle codifiche dipendenti dallo stato. Se `s` punta al carattere null, la funzione restituisce il valore zero. Negli altri casi la funzione restituisce il numero di byte presenti nel carattere multibyte puntato da `s`. Viene restituito il valore -1 se i prossimi `n` o meno byte non formano un carattere multibyte valido.

25.2

memchr	<i>Ricerca un carattere in un blocco di memoria</i>	<string.h>
<code>void *memchr(const void *s, int c, size_t n);</code>		
<i>Restituisce</i>		
Un puntatore alla prima occorrenza del carattere <i>c</i> tra i primi <i>n</i> caratteri dell'oggetto puntato da <i>s</i> . Se il carattere <i>c</i> non viene trovato restituisce un puntatore nullo.		
23.6		
<hr/>		
memcmp	<i>Confronto di un blocco di memoria</i>	<string.h>
<code>int memcmp(const void *s1, const void *s2, size_t n);</code>		
<i>Restituisce</i>		
Un intero negativo, pari a zero o positivo a seconda che i primi <i>n</i> caratteri dell'oggetto puntato da <i>s1</i> siano minori, uguali o maggiori ai primi <i>n</i> caratteri dell'oggetto puntato da <i>s2</i> .		
23.6		
<hr/>		
memcpy	<i>Copia di un blocco di memoria</i>	<string.h>
<code>void *memcpy(void * restrict s1, const void * restrict s2, size_t n);</code>		
Copia <i>n</i> caratteri dall'oggetto puntato da <i>s2</i> nell'oggetto puntato da <i>s1</i> . Il comportamento non è definito se gli oggetti si sovrappongono.		
<i>Restituisce</i>		
<i>s1</i> (un puntatore alla destinazione).		
23.6		
<hr/>		
memmove	<i>Copia di un blocco di memoria</i>	<string.h>
<code>void *memmove(void *s1, const void *s2, size_t n);</code>		
Copia <i>n</i> caratteri dall'oggetto puntato da <i>s2</i> nell'oggetto puntato da <i>s1</i> . La funzione opera correttamente anche se gli oggetti si sovrappongono.		
<i>Restituisce</i>		
<i>s1</i> (un puntatore alla destinazione).		
23.6		
<hr/>		
memset	<i>Inizializzazione di un blocco di memoria</i>	<string.h>
<code>void *memset(void *s, int c, size_t n);</code>		
Salva <i>c</i> in ognuno dei primi <i>n</i> caratteri dell'oggetto puntato da <i>s</i> .		
<i>Restituisce</i>		
<i>s</i> (un puntatore all'oggetto).		
23.6		
<hr/>		
mktime	<i>Conversione di un ora locale del tipo broken down in una del tipo calendar time</i>	<time.h>
<code>time_t mktime(struct tm *timeptr);</code>		
Converte un ora locale del tipo broken down (contenuta nella struttura puntata da <i>timeptr</i>) nel formato <i>calendar time</i> . I membri della struttura non devono essere necessariamente all'interno dei rispettivi intervalli di valori ammissibili. Inoltre vengono ignorati i valori di <i>tm_wday</i> (giorno della settimana) e <i>tm_yday</i> (giorno dell'anno). La funzione salva i valori di <i>tm_wday</i> e <i>tm_yday</i> dopo aver regolato gli altri membri portandoli all'interno dei rispettivi intervalli.		
<i>Restituisce</i>		
Un'ora nel formato <i>calendar time</i> corrispondente alla struttura puntata da <i>timeptr</i> . Restituisce il valore (<i>time_t</i>) (-1) se l'ora nel formato <i>calendar time</i> non può essere rappresentata.		
26.3		
<hr/>		
modf	<i>Suddivisione in parte intera e frazionaria</i>	<math.h>
<code>double modf(double value, double *iptr);</code>		
modff		
<code>float modff(float value, float *iptr);</code>		
modfl		
<code>long double modfl(long double value, long double *iptr);</code>		
Suddivide <i>value</i> nelle parti intera e frazionaria. Salva la parte intera nell'oggetti puntato da <i>iptr</i> .		
<i>Restituisce</i>		
Parte frazionaria di <i>value</i> .		
23.3		
<hr/>		
nan	<i>Creazione di un NaN (C99)</i>	<math.h>
<code>double nan(const char *tagp);</code>		

nanf

```
float nanf(const char *tagp);
```

nanl

```
long double nanl(const char *tagp);
```

Restituisce

Un NaN “tranquillo” il cui pattern binario è determinato dalla stringa puntata da `tagp`. Restituisce il valore zero se gli NaN tranquilli non sono supportati. 23.4

nearbyint Arrotondamento a un valore integrale usando la direzione corrente (C99) <math.h>

```
double nearbyint(double x);
```

nearbyintf

```
float nearbyintf(float x);
```

nearbyintl

```
long double nearbyintl(long double x);
```

Restituisce

`x` arrotondato a un intero (nel formato a virgola mobile) usando la direzione di arrotondamento corrente. Non solleva l’eccezione floating point *inexact*. 23.4

nextafter Numero successivo dopo (C99) <math.h>

```
double nextafter(double x, double y);
```

nextafterf

```
float nextafterf(float x, float y);
```

nextafterl

```
long double nextafterl(long double x, long double y);
```

Restituisce

Il successivo valore rappresentabile dopo `x` nella direzione di `y`. Restituisce il valore immediatamente precedente a `x` se `y < x` oppure il valore immediatamente successivo a `x` se `x < y`. Restituisce `y` se `x` è uguale a `y`. Se il valore assoluto di `x` corrisponde al più grande valore rappresentabile e il risultato è infinito o non rappresentabile può verificarsi un errore di intervallo. 23.4

nexttoward Numero successivo verso (C99) <math.h>

```
double nexttoward(double x, long double y);
```

nexttowardf

```
float nexttowardf(float x, long double y);
```

nexttowardl

```
long double nexttowardl(long double x, long double y);
```

Restituisce

Il successivo valore rappresentabile dopo `x` nella direzione di `y` (vedi `nextafter`). Se `x` è uguale a `y` restituisce `y` convertito al tipo della funzione. 23.4

perror Stampa di un messaggio di errore <stdio.h>

```
void perror(const char *s);
```

Scrive il seguente messaggio nello stream `stderr`:

stringa: messaggio-di-errore

stringa è la stringa puntata da `s` mentre *messaggio-di-errore* è un messaggio definito dall’implementazione che corrisponde a quello restituito dalla chiamata `strerror(errno)`. 24.2

pow Potenza <math.h>

```
double pow(double x, double y);
```

powf

```
float powf(float x, float y);
```

powl

```
long double powl(long double x, long double y);
```

Restituisce

`x` elevato alla potenza `y`. In certi casi (che cambiano dal C89 al C99) può verificarsi un errore di dominio o di intervallo. 23.3

printf Scrittura formattata <stdio.h>

```
int printf(const char * restrict format, ...);
```

Scrivere dell'output nello stream `stdout`. La stringa puntata da `format` specifica come verranno visualizzati gli argomenti seguenti.

Restituisce

Il numero di caratteri scritti. Se si verifica un errore restituisce un valore negativo. 3.1, 22.3

putc *Scrittura di un carattere su file* <stdio.h>

```
int putc(int c, FILE *stream);
```

Scrivere il carattere `c` nello stream puntato da `stream`. *Nota:* normalmente viene implementata come una macro e può valutare stream più di una volta.

Restituisce

`c` (il carattere scritto). Se si verifica un errore di scrittura, la funzione imposta l'indicatore di errore associato allo stream e restituisce il valore EOF. 22.4

putchar *Scrittura di un carattere* <stdio.h>

```
int putchar(int c);
```

Scrivere il carattere `c` nello stream `stdout`. *Nota:* normalmente viene implementata come una macro.

Restituisce

`c` (il carattere scritto). Se si verifica un errore di scrittura, la funzione imposta l'indicatore di errore associato allo stream e restituisce il valore EOF. 7.3, 22.4

puts *Scrittura di una stringa* <stdio.h>

```
int puts(const char *s);
```

Scrivere la stringa puntata da `s` nello stream `stdout` e poi scrivere il carattere new-line.

Restituisce

Un valore non negativo se ha successo, EOF se si verifica un errore nella scrittura. 13.3, 22.5

putwc *Scrittura su file di un wide character (C99)* <wchar.h>

```
wint_t putwc(wchar_t c, FILE *stream);
```

Versione wide character della `putc`. 25.5

putwchar *Write Wide Character (C99)* <wchar.h>

```
wint_t putwchar(wchar_t c);
```

Versione wide character della `putchar`. 25.5

qsort *Ordinamento di un vettore* <stdlib.h>

```
void qsort(void *base, size_t nmemb, size_t size,
           int (*compar)(const void *, const void *));
```

Ordina il vettore puntato da `base`. Il vettore possiede `nmemb` elementi, ognuno lungo `size` byte. L'argomento `compar` è un puntatore a una funzione di confronto. Quando le viene passato un puntatore a due elementi del vettore, la funzione di confronto deve restituire un intero negativo, pari a zero o positivo a seconda che il primo elemento sia minore, uguale o maggiore del secondo. 17.7, 26.2

raise *Generazione di un segnale* <signal.h>

```
int raise(int sig);
```

Genera un segnale il cui numero è `sig`.

Restituisce

Il valore zero se ha successo, un valore diverso da zero in caso contrario. 24.3

rand *Generazione di numeri pseudo casuali* <stdlib.h>

```
int rand(void);
```

Restituisce

Un intero pseudo casuale compreso tra 0 e `RAND_MAX` (incluso). 26.2

realloc *Ridimensionamento di un blocco di memoria* <stdlib.h>

```
void *realloc(void *ptr, size_t size);
```

`ptr` è assunto puntare a un blocco di memoria precedentemente ottenuto dalla funzioni `calloc`, `malloc` o `realloc`. La funzione alloca un blocco di `size` byte, copiando, se necessario, il contenuto del vecchio blocco.

Restituisce

Un puntatore all'inizio del nuovo blocco. Restituisce un puntatore nullo se non può essere allocato un blocco delle dimensioni richieste. 17.3

remainder *Resto (C99)* <math.h>

```
double remainder(double x, double y);
```

remainderf

```
float remainderf(float x, float y);
```

remainderl

```
long double remainderl(long double x, long double y);
```

Restituisce
 $x - ny$, dove n è l'intero più prossimo al valore esatto di x/y (Se x/y è a metà strada tra due interi, n è pari). Se $x - ny = 0$, il valore restituito ha lo stesso segno di x . Se y è uguale a zero, può verificarsi un errore di dominio oppure può venire restituito il valore zero. 23.4

remove *Rimozione di un file* <stdio.h>

```
int remove(const char *filename);
```

Cancella il file il cui nome è puntato da `filename`.

Restituisce
 Il valore zero se ha successo, un valore diverso da zero altrimenti. 22.2

remquo *Resto e quoziente (C99)* <math.h>

```
double remquo(double x, double y, int *quo);
```

remquof

```
float remquof(float x, float y, int *quo);
```

remquol

```
long double remquol(long double x, long double y,
                    int *quo);
```

Calcola sia il resto che il quoziente ottenuto dividendo x per y . L'oggetto puntato da `quo` viene modificato in modo che contenga gli n bit di ordine inferiore del quoziente intero $|x/y|$, dove n è definito dall'implementazione ma deve essere almeno pari a tre. Il valore salvato in questo oggetto sarà negativo se $x/y < 0$.

Restituisce
 Lo stesso valore della corrispondente funzione `remainder`. Se y è uguale a zero, può verificarsi un errore di dominio oppure può venir restituito il valore zero. 23.4

rename *Rinominare un file* <stdio.h>

```
int rename(const char *old, const char *new);
```

Cambia il nome di un file. I parametri `old` e `new` puntano a delle stringhe contenenti rispettivamente il vecchio e il nuovo nome.

Restituisce
 Il valore zero se l'operazione di rinomina è andata a buon fine. Se l'operazione fallisce (magari a causa del fatto che il vecchio file è attualmente aperto) restituisce un valore diverso da zero. 22.2

rewind *Riavvolgimento di un file* <stdio.h>

```
void rewind(FILE *stream);
```

Imposta l'indicatore della posizione del file per lo stream puntato da `stream` all'inizio del file stesso. Azzeri gli indicatori di errore e di end-of-file associati allo stream. 22.7

rint *Arrotondamento a un valore integrale usando la direzione corrente (C99)* <math.h>

```
double rint(double x);
```

rintf

```
float rintf(float x);
```

rintl

```
long double rintl(long double x);
```

Restituisce
 x arrotondato a un intero (in formato a virgola mobile) usando la direzione di arrotondamento corrente. Se il valore ha un risultato diverso da x la funzione può sollevare l'eccezione floating point *inexact*. 23.4

<p>round <i>Arrotondamento al valore integrale più vicino (C99)</i></p> <pre>double round(double x);</pre> <p><i>float roundf(float x);</i></p> <p><i>long double roundl(long double x);</i></p> <p><i>Restituisce</i> <i>x arrotondato all'intero più vicino (nel formato a virgola mobile). I casi a metà strada vengono arrotondati allontanandosi dallo zero.</i></p>	<p><math.h></p> <p>roundf</p> <p>roundl</p> <p>23.4</p>
<p>scalbln <i>Scalare un numero virgola mobile usando un long int (C99)</i></p> <pre>double scalbln(double x, long int n);</pre> <p><i>float scalblnf(float x, long int n);</i></p> <p><i>long double scalblnl(long double x, long int n);</i></p> <p><i>Restituisce</i> <i>$x \times \text{FLT_RADIX}^n$, calcolato in un modo efficiente. Può verificarsi un errore di intervallo.</i></p>	<p><math.h></p> <p>scalblnf</p> <p>scalblnl</p> <p>23.4</p>
<p>scalbn <i>Scalare un numero virgola mobile usando un intero (C99)</i></p> <pre>double scalbn(double x, int n);</pre> <p><i>float scalbnf(float x, int n);</i></p> <p><i>long double scalbnl(long double x, int n);</i></p> <p><i>Restituisce</i> <i>$x \times \text{FLT_RADIX}^n$, calcolato in un modo efficiente. Può verificarsi un errore di intervallo.</i></p>	<p><math.h></p> <p>scalbnf</p> <p>scalbnl</p> <p>23.4</p>
<p>scanf <i>Lettura formattata</i></p> <pre>int scanf(const char * restrict format, ...);</pre> <p>Legge degli oggetti di input dallo stream <code>stdin</code>. La stringa puntata da <code>format</code> specifica il formato degli oggetti che devono essere letti. Gli argomenti che seguono <code>format</code> puntano agli oggetti nei quali i dati letti devono essere salvati.</p> <p><i>Restituisce</i> Il numero degli oggetti di input letti e salvati con successo. Se si verifica un errore prima che qualsiasi oggetto possa essere letto restituisce EOF.</p>	<p><stdio.h></p> <p>3.2, 22.3</p>
<p>setbuf <i>Impostazione del buffer</i></p> <pre>void setbuf(FILE * restrict stream, char * restrict buf);</pre> <p>Se <code>buf</code> non è un puntatore nullo, una chiamata alla funzione è equivalente a:</p> <pre>(void) setvbuf(stream, buf, _IOFBF, BUFSIZ);</pre> <p>altrimenti è equivalente a:</p> <pre>(void) setvbuf(stream, NULL, _IONBF, 0);</pre>	<p><stdio.h></p> <p>22.2</p>
<p>setjmp <i>Predisposizione per un salto non locale</i></p> <pre>int setjmp(jmp_buf env);</pre> <p>Salva l'ambiente corrente in <code>env</code> perché possa essere usato in una successiva chiamata alla <code>longjmp</code>.</p> <p><i>Restituisce</i> Il valore zero quando viene chiamata direttamente. Un valore diverso da zero quando avviene il ritorno da una chiamata alla <code>longjmp</code>.</p>	<p><setjmp.h></p> <p><i>macro</i></p> <p>24.4</p>
<p>setlocale <i>Impostazione della localizzazione</i></p> <pre>char *setlocale(int category, const char *locale);</pre> <p>Imposta una porzione della localizzazione del programma. L'argomento <code>category</code> indica quale porzione viene coinvolta. L'argomento <code>locale</code> punta a una stringa rappresentante la nuova localizzazione.</p> <p><i>Restituisce</i></p>	<p><locale.h></p>

Se `locale` è un puntatore nullo, la funzione restituisce un puntatore alla stringa associata a `category` nella localizzazione corrente. Negli altri casi la funzione restituisce un puntatore alla stringa associata a `category` nella nuova localizzazione. Se l'operazione fallisce restituisce un puntatore nullo. 25.1

setvbuf *Impostazione del buffer*

<stdio.h>

```
int setvbuf(FILE * restrict stream,
            char * restrict buf,
            int mode, size_t size);
```

Modifica il buffering dello stream puntato da `stream`. Il valore di `mode` può essere `_IOFBF` (*full buffering*), `_IOLBF` (*line buffering*) o `_IONBF` (*no buffering*). Se `buf` è un puntatore nullo, se è necessario viene allocato automaticamente un buffer. Altrimenti `buf` punta a un blocco di memoria che può essere usato come buffer, `size` è il numero di byte presenti nel blocco. *Nota*: la funzione deve essere chiamata dopo che lo stream è stato aperto ma prima che venga eseguita qualsiasi operazione su di esso.

Restituisce

Il valore zero se l'operazione ha successo. Un valore diverso da zero se `mode` non è valido o se la richiesta non può essere soddisfatta. 22.2

signal *Installazione di un handler di segnale*

<signal.h>

```
void (*signal(int sig, void (*func)(int)))(int);
```

Installa la funzione puntata da `func` come handler per il segnale il cui numero è `sig`. Passare `SIG_DFL` come secondo argomento fa sì che venga impiegato l'handler di default mentre passare `SIG_IGN` fa sì che il segnale venga ignorato.

Restituisce

Un puntatore precedente all'handler per questo segnale. Se l'handler non può essere installato restituisce `SIG_ERR` e salva un valore positivo in `errno`. 24.3

signbit *Bit di segno (C99)*

<math.h>

```
int signbit(real-floating x); macro
```

Restituisce

Un valore diverso da zero se il segno di `x` è negativo e pari a zero altrimenti. Il valore di `x` può essere un numero qualsiasi, incluso infinito e NaN. 23.4

sin *Seno*

<math.h>

```
double sin(double x);
```

sinf

```
float sinf(float x);
```

sinl

```
long double sinl(long double x);
```

Restituisce

Il seno di `x` (misurato in radianti).

23.3

sinh *Seno iperbolico*

<math.h>

```
double sinh(double x);
```

sinhf

```
float sinhf(float x);
```

sinhl

```
long double sinhl(long double x);
```

Restituisce

Il seno iperbolico di `x`. Se il valore assoluto di `x` è troppo grande si verifica un errore di intervallo.

23.3

snprintf *Scrittura formattata su stringa limitata(C99)*

<stdio.h>

```
int snprintf(char * restrict s, size_t n,
             const char * restrict format, ...);
```

Equivalente alla `fprintf`, ma salva i caratteri nel vettore puntato da `s` invece che scriverli su uno stream. Nel vettore non verranno scritti più di `n - 1` caratteri. La stringa puntata da `format` specifica come debbano essere visualizzati gli argomenti seguenti. Alla fine dell'output salva il carattere null all'interno del vettore.

Restituisce

Il numero di caratteri che sarebbero stati salvati nel vettore (senza includere il carattere null) se non ci fosse stata nessuna restrizione di lunghezza. Se si verifica un errore di codifica restituisce un valore negativo. 22.8

sprintf	<i>Scrittura formattata su stringa</i>	<stdio.h>
<pre>int sprintf(char * restrict s, const char * restrict format, ...);</pre>		
<p>Equivalente alla <code>fprintf</code> ma salva i caratteri nel vettore puntato da <code>s</code> invece che scriverli su uno stream. La stringa puntata da <code>format</code> specifica come debbano essere visualizzati gli argomenti seguenti. Alla fine dell'output scrive nel vettore il carattere null.</p>		
<i>Restituisce</i>		
<p>Il numero di caratteri salvati nel vettore, non includendo il carattere null. Nel C99 restituisce un valore negativo qualora si verificasse un errore di codifica.</p>		
22.8		
sqrt	<i>Radice quadrata</i>	<math.h>
<pre>double sqrt(double x);</pre>		
<i>sqrtf</i>		
<pre>float sqrtf(float x);</pre>		
<i>sqrtl</i>		
<pre>long double sqrtl(long double x);</pre>		
<i>Restituisce</i>		
<p>La radice quadrata non negativa di <code>x</code>. Se <code>x</code> è negativo si verifica un errore di dominio.</p>		
23.3		
srand	<i>Seme per il generatore di numeri pseudocasuali</i>	<stdlib.h>
<pre>void srand(unsigned int seed);</pre>		
<p>Utilizza <code>seed</code> per inizializzare la sequenza di numeri pseudo casuali prodotta chiamando la funzione <code>rand</code>.</p>		
26.2		
sscanf	<i>Lettura formattata da stringa</i>	<stdio.h>
<pre>int sscanf(const char * restrict s, const char * restrict format, ...);</pre>		
<p>Equivalente alla <code>fscanf</code> ma legge i caratteri dalla stringa puntata da <code>s</code> invece che leggerli da uno stream. La stringa puntata da <code>format</code> specifica il formato degli oggetti che devono essere letti. Gli argomenti che seguono <code>format</code> puntano agli oggetti nei quali i dati letti devono essere salvati.</p>		
<i>Restituisce</i>		
<p>Il numero di oggetti di input che sono stati letti e salvati con successo. Se l'input fallisce prima che possa essere letto qualsiasi oggetto la funzione restituisce EOF.</p>		
22.8		
strcat	<i>Concatenamento di una stringa</i>	<string.h>
<pre>char *strcat(char * restrict s1, const char * restrict s2);</pre>		
<p>Accoda i caratteri della stringa puntata da <code>s2</code> nella stringa puntata da <code>s1</code>.</p>		
<i>Restituisce</i>		
<p><code>s1</code> (un puntatore alla stringa concatenata).</p>		
13.5, 23.6		
strchr	<i>Ricerca di un carattere in una stringa</i>	<string.h>
<pre>char *strchr(const char *s, int c);</pre>		
<i>Restituisce</i>		
<p>Un puntatore alla prima occorrenza del carattere <code>c</code> nella stringa puntata da <code>s</code>. Se il carattere <code>c</code> non viene trovato restituisce un puntatore nullo.</p>		
23.6		
strcmp	<i>Confronto tra stringhe</i>	<string.h>
<pre>int strcmp(const char *s1, const char *s2);</pre>		
<i>Restituisce</i>		
<p>Un intero negativo, pari a zero o positivo a seconda che la stringa puntata da <code>s1</code> sia minore, uguale o maggiore di quella puntata da <code>s2</code>.</p>		
13.5, 23.6		
strcoll	<i>Confronto tra stringhe usando una sequenza di confronto specifica della localizzazione</i>	<string.h>
<pre>int strcoll(const char *s1, const char *s2);</pre>		
<i>Restituisce</i>		
<p>Un intero negativo, pari a zero o positivo a seconda che la stringa puntata da <code>s1</code> sia minore, uguale o maggiore di quella puntata da <code>s2</code>. Il confronto viene eseguito seguendo le regole della categoria <code>LC_COLLATE</code> della localizzazione corrente.</p>		
23.6		

<p>strcpy <i>Copia di una stringa</i></p> <pre>char *strcpy(char * restrict s1, const char * restrict s2);</pre> <p>Copia la stringa puntata da <code>s2</code> nel vettore puntato da <code>s1</code>.</p> <p><i>Restituisce</i> <code>s1</code> (un puntatore alla destinazione).</p>	<p><string.h></p> <p style="text-align: right;">13.5, 23.6</p>
<p>strcspn <i>Ricerca in una stringa di uno span iniziale di caratteri non appartenenti a un set</i></p> <pre>size_t strcspn(const char *s1, const char *s2);</pre> <p><i>Restituisce</i> La lunghezza del più lungo segmento della stringa puntata da <code>s1</code> che non contiene nessuno dei caratteri presenti nella stringa puntata da <code>s2</code>.</p>	<p><string.h></p> <p style="text-align: right;">23.6</p>
<p>strerror <i>Conversione di un numero di errore in stringa</i></p> <pre>char *strerror(int errnum);</pre> <p><i>Restituisce</i> Un puntatore a una stringa contenente il messaggio di errore corrispondente al valore <code>errnum</code>.</p>	<p><string.h></p> <p style="text-align: right;">24.2</p>
<p>strftime <i>Scrittura formattata di data e ora su stringa</i></p> <pre>size_t strftime(char * restrict s, size_t maxsize, const char * restrict format, const struct tm * restrict timeptr);</pre> <p>Salva i caratteri nel vettore puntato da <code>s</code> sotto il controllo della stringa puntata da <code>format</code>. Il formato della stringa può contenere caratteri ordinari, i quali vengono copiati senza essere modificati, e specificatori di conversione, i quali vengono sostituiti dai validi presi dalla struttura puntata da <code>timeptr</code>. Il parametro <code>maxsize</code> limita il numero di caratteri (incluso il carattere null) che possono essere salvati.</p> <p><i>Restituisce</i> Il numero di caratteri salvati (senza includere il carattere null di fine). Restituisce il valore zero se il numero di caratteri che devono essere salvati (incluso il carattere null) eccede il limite <code>maxsize</code>.</p>	<p><time.h></p> <p style="text-align: right;">26.3</p>
<p>strlen <i>Lunghezza di una stringa</i></p> <pre>size_t strlen(const char *s);</pre> <p><i>Restituisce</i> La lunghezza della stringa puntata da <code>s</code> senza includere il carattere null.</p>	<p><string.h></p> <p style="text-align: right;">13.5, 23.6</p>
<p>strncat <i>Concatenamento di stringhe limitato</i></p> <pre>char *strncat(char * restrict s1, const char * restrict s2, size_t n);</pre> <p>Accoda i caratteri del vettore puntato da <code>s2</code> alla stringa puntata da <code>s1</code>. La copia si interrompe quando viene incontrato un carattere null oppure quando sono stati copiati <code>n</code> caratteri.</p> <p><i>Restituisce</i> <code>s1</code> (un puntatore alla stringa concatenata).</p>	<p><string.h></p> <p style="text-align: right;">13.5, 23.6</p>
<p>strncmp <i>Confronto tra stringhe limitato</i></p> <pre>int strncmp(const char *s1, const char *s2, size_t n);</pre> <p><i>Restituisce</i> Un intero negativo, pari a zero o positivo a seconda che i primi <code>n</code> caratteri del vettore puntato da <code>s1</code> siano minori, uguali o maggiori dei primi <code>n</code> caratteri puntati da <code>s2</code>. Il confronto si arrasta quando in uno dei due vettori viene incontrato un carattere null.</p>	<p><string.h></p> <p style="text-align: right;">23.6</p>
<p>strncpy <i>Copia limitata di una stringa</i></p> <pre>char *strncpy(char * restrict s1, const char * restrict s2, size_t n);</pre> <p>Copia i primi <code>n</code> caratteri del vettore puntato da <code>s2</code> nel vettore puntato da <code>s1</code>. Se incontra un carattere null nel vettore puntato da <code>s2</code>, la funzione aggiunge dei caratteri null nel vettore puntato da <code>s1</code> fino a quando non sono stati scritti un totale di <code>n</code> caratteri.</p> <p><i>Restituisce</i></p>	<p><string.h></p>

`s1` (un puntatore alla destinazione).

13.5, 23.6

strupbrk *Ricerca in una stringa di un carattere appartenente a un set* <string.h>

```
char *strupbrk(const char *s1, const char *s2);
```

Restituisce

Un puntatore al carattere più a sinistra presente nella stringa puntata da `s1` che combaci con un qualsiasi carattere presente nella stringa puntata da `s2`. Se non viene trovata nessuna corrispondenza restituisce un puntatore nullo. 23.6

strrchr *Ricerca di un carattere all'interno di una stringa* <string.h>

seguendo l'ordine inverso

```
char *strrchr(const char *s, int c);
```

Restituisce

Un puntatore all'ultima occorrenza del carattere `c` nella stringa puntata da `s`. Se il carattere `c` non viene trovato restituisce un puntatore nullo. 23.6

strspn *Ricerca in una stringa di uno span iniziale di caratteri* <string.h>

appartenenti a un set

```
size_t strspn(const char *s1, const char *s2);
```

Restituisce

La lunghezza del più lungo segment iniziale della stringa puntata da `s1` che consista interamente di caratteri presenti nella stringa puntata da `s2`. 23.6

strstr *Ricerca di una sottostringa all'interno di una stringa* <string.h>

```
char *strstr(const char *s1, const char *s2);
```

Restituisce

Un puntatore alla prima occorrenza nella stringa puntata da `s1` della sequenza di caratteri presente nella stringa puntata da `s2`. Se non viene trovata nessuna occorrenza restituisce un puntatore nullo. 23.6

strtod *Conversione di una stringa in un double* <stdlib.h>

```
double strtod(const char * restrict nptr,
              char ** restrict endptr);
```

Salta i caratteri di spazio bianco presenti nella stringa puntata da `nptr` e poi converte i caratteri seguenti in un valore `double`. Se `endptr` non è un puntatore nullo, la `strtod` modifica l'oggetto puntato da questo in modo che punti al primo carattere residuo. Se non viene trovato nessun valore `double`, o se questo non è nel formato corretto, la funzione salva `nptr` nell'oggetto puntato da `endptr`. Se il numero è troppo grande o troppo piccolo per essere rappresentato la funzione salva il valore `ERANGE` nella variabile `errno`. *Modifica del C99*: la stringa puntata da `nptr` può contenere un numero a virgola mobile esadecimale, infinito o NaN. Se `ERANGE` venga salvato in `errno` quando il numero è troppo piccolo per essere rappresentato dipende dall'implementazione.

Restituisce

Il numero convertito. Restituisce il valore zero se la conversione non può essere eseguita. Se il numero è troppo grande per essere rappresentato, restituisce il valore `HUGE_VAL` con segno più o con il segno meno a seconda di quello del numero. Restituisce il valore zero se il numero è troppo piccolo per essere rappresentato. *Modifica del C99*: se il numero è troppo piccolo per essere rappresentato la funzione restituisce un valore il cui valore assoluto non è maggiore del più piccolo numero normalizzato positivo di tipo `double`. 26.2

strtof *Conversione di una stringa in un float (C99)* <stdlib.h>

```
float strtof(const char * restrict nptr,
            char ** restrict endptr);
```

La funzione è identica alla `strtod` ad eccezione del fatto che converte la stringa in un valore `float`.

Restituisce

Il numero convertito. Restituisce il valore zero se la conversione non può essere eseguita. Se il numero è troppo grande per essere rappresentato, la funzione restituisce il valore `HUGE_VALF` con segno più o con il segno meno a seconda di quello del numero. Se il numero è troppo piccolo per essere rappresentato, la funzione restituisce un valore il cui valore assoluto non è maggiore del più piccolo valore normalizzato positivo di tipo `float`. 26.2

strtoumax *Conversione di una stringa all'intero della dimensione più grande (C99)* <inttypes.h>

```
intmax_t strtoumax(const char * restrict nptr,
                  char ** restrict endptr, int base);
```

La funzione è identica alla `strtol` ma a differenza di questa converte una stringa in un valore di tipo `intmax_t` (il tipo intero con segno della dimensione più grande).

Restituisce

Il numero convertito. Restituisce il valore zero se la conversione non può essere eseguita. Se il numero non può essere rappresentato restituisce `INTMAX_MAX` o `INTMAX_MIN` a seconda del segno del numero. 27.2

`strtok` *Ricerca di un token in una stringa* <string.h>

```
char *strtok(char * restrict s1,
             const char * restrict s2);
```

Cerca nella stringa puntata da `s1` un “token” formato da caratteri non presenti nella stringa puntata da `s2`. Se il token esiste, il carattere che lo segue viene tramutato nel carattere nullo. Se `s1` è un puntatore nullo viene continuata la ricerca iniziata dalla più recente chiamata alla `strtok`. In tal caso la ricerca inizia immediatamente dopo il carattere nullo presente alla fine del token precedente.

Restituisce

Un puntatore al primo carattere del token. Se il token non viene trovato la funzione restituisce un puntatore nullo. 23.6

`strtol` *Conversione di una stringa in long int* <stdlib.h>

```
long int strtol(const char * restrict nptr,
               char ** restrict endptr, int base);
```

Salta i caratteri di spazio bianco nella stringa puntata da `nptr` e poi converte i caratteri seguenti in un valore `long int`. Se `base` è compreso tra 2 e 36, viene usato come radice del numero. Se `base` è uguale a zero il numero viene assunto essere decimale a meno che non inizi con lo 0 (ottale) oppure con 0x o 0X (esadecimale). Se `endptr` non è un puntatore nullo, la funzione modifica l’oggetto puntato da `endptr` in modo che punti al primo carattere residuo. Se non viene trovato nessun valore `long int` o se non è della forma corretta, la funzione salva `nptr` nell’oggetto puntato da `endptr`. Se il numero non può essere rappresentato la funzione salva il valore `ERANGE` nella variabile `errno`.

Restituisce

Il numero convertito. Se la conversione non può essere eseguita restituisce il valore zero. Se il numero non può essere rappresentato restituisce il valore `LONG_MAX` o `LONG_MIN` a seconda del segno del numero. 26.2

`strtold` *Conversione di una stringa in un long double (C99)* <stdlib.h>

```
long double strtold(const char * restrict nptr,
                   char ** restrict endptr);
```

La funzione è identica alla `strtod` ma a differenza di questa converte una stringa in un valore `long double`.

Restituisce

Il numero convertito. Se la conversione non può essere eseguita restituisce il valore zero. Se il numero è troppo grande per essere rappresentato la funzione restituisce il valore `HUGE_VALL` con il segno più o con il segno meno a seconda di quello del numero. Se il numero è troppo piccolo per essere rappresentato, la funzione restituisce un valore il cui valore assoluto non è maggiore di quello del più piccolo valore normalizzato positivo del tipo `long double`. 26.2

`strtoll` *Conversione di una stringa in un long long int (C99)* <stdlib.h>

```
long long int strtoll(const char * restrict nptr,
                    char ** restrict endptr,
                    int base);
```

La funzione è identica alla `strtol` ma differisce da questa per il fatto che converte una stringa in un valore `long long int`.

Restituisce

Il numero convertito. Se la conversione non può essere eseguita restituisce il valore zero. Se il numero non può essere rappresentato la funzione restituisce il valore `LLONG_MAX` o `LLONG_MIN` a seconda del segno del numero. 26.2

`strtoul` *Conversione di una stringa in un unsigned long int* <stdlib.h>

```
unsigned long int strtoul(const char * restrict nptr,
                        char ** restrict endptr,
                        int base);
```

La funzione è identica alla `strtol` ma differisce da questa per il fatto che converte una stringa in un valore `unsigned long int`.

Restituisce

Il numero convertito. Se la conversione non può essere eseguita restituisce il valore zero. Se il numero non può essere

rappresentato la funzione restituisce il valore `ULONG_MAX`.

26.2

strtoull *Conversione di una stringa in un unsigned long long int (C99)* <stdlib.h>

```
unsigned long long int strtoull(
    const char * restrict nptr,
    char ** restrict endptr, int base);
```

La funzione è identica alla `strtoul` ma differisce da questa per il fatto che converte una stringa in un valore `unsigned long long int`.

Restituisce

Il numero convertito. Se la conversione non può essere eseguita la funzione restituisce il valore zero. Se il numero non può essere rappresentato la funzione restituisce il valore `ULLONG_MAX`.

26.2

strtoumax *Conversione di una stringa in un intero senza segno della <inttypes.h> dimensione più grande (C99)*

```
uintmax_t strtoumax(const char * restrict nptr,
    char ** restrict endptr,
    int base);
```

La funzione è identica alla `strtoul` ma differisce da questa per il fatto che converte una stringa in un valore `uintmax_t` (il tipo intero senza segno della dimensione più grande).

Restituisce

Il numero convertito. Se la conversione non può essere eseguita la funzione restituisce il valore zero. Se il numero non può essere rappresentato la funzione restituisce il valore `UINTMAX_MAX`.

27.2

strxfrm *Trasformazione di una stringa* <string.h>

```
size_t strxfrm(char * restrict s1,
    const char * restrict s2, size_t n);
```

Trasforma la stringa puntata da `s2` ponendo i primi `n` caratteri del risultato (incluso il carattere null) nel vettore puntato da `s1`. Una chiamata alla `strcmp` con due stringhe trasformate deve produrre lo stesso risultato (negativo, pari a zero o negativo) che chiamare la funzione `strcmp` con le stringhe originali. Se `n` è uguale a zero, viene ammesso che `s1` sia un puntatore nullo.

Restituisce

La lunghezza della stringa trasformata. Se questo valore è `n` o più, il contenuto del vettore puntato da `s1` è indeterminato.

23.6

swprintf *Scrittura formattata di wide character su stringa (C99)* <wchar.h>

```
int swprintf(wchar_t * restrict s, size_t n,
    const wchar_t * restrict format, ...);
```

Equivalente alla `fprintf` ma salva i wide character nel vettore puntato da `s` invece che scriverli in uno stream. La stringa puntata da `format` specifica come debbano essere visualizzati gli argomenti seguenti. Nel vettore non verranno scritti più di `n` wide character, incluso il wide character null.

Restituisce

Il numero di wide character salvati nel vettore, senza includere il wide character null. Restituisce un valore negativo se si verifica un errore di codifica o se il numero di wide character che devono essere scritti è pari a `n` o più.

25.5

swscanf *Lettura formattata di wide character da stringa (C99)* <wchar.h>

```
int swscanf(const wchar_t * restrict s,
    const wchar_t * restrict format, ...);
```

Versione wide character della `sscanf`.

25.5

system *Esecuzione di un comando del sistema operativo* <stdlib.h>

```
int system(const char *string);
```

Passa la stringa puntata da `string` al processore di comandi del sistema operativo (shell) in modo che venga eseguita. Come risultato dell'esecuzione del comando il programma può terminare.

Restituisce

Se `string` è un puntatore nullo la funzione restituisce un valore diverso da zero nel caso in cui sia disponibile un processore di comandi. Se la stringa non è un puntatore nullo, la funzione restituisce un valore definito dall'implementazione (ammesso che la funzione restituisce il controllo).

26.2

tan <i>Tangente</i>	<math.h>
<code>double tan(double x);</code>	tanf
<code>float tanf(float x);</code>	tanl
<code>long double tanl(long double x);</code>	
<i>Restituisce</i>	
La tangente di x (misurata in radianti).	23.3

tanh <i>Tangente iperbolica</i>	<math.h>
<code>double tanh(double x);</code>	tanhf
<code>float tanhf(float x);</code>	tanh1
<code>long double tanh1(long double x);</code>	
<i>Restituisce</i>	
La tangente iperbolica di x .	23.3

tgamma <i>Funzione gamma (C99)</i>	<math.h>
<code>double tgamma(double x);</code>	tgammaf
<code>float tgammaf(float x);</code>	tgammal
<code>long double tgammal(long double x);</code>	
<i>Restituisce</i>	
$\Gamma(x)$, dove Γ è la funzione gamma. Se x è un intero negativo o pari a zero può verificarsi un errore di dominio o di intervallo. Se il valore assoluto di x è troppo grande o troppo piccolo può verificarsi un errore di intervallo.	23.4

time <i>Ora corrente</i>	<time.h>
<code>time_t time(time_t *timer);</code>	
<i>Restituisce</i>	
L'ora corrente nel formato calendar time. Restituisce (time_t) (-1) se l'ora calendar time non è disponibile. Inoltre se timer non è un puntatore nullo, la funzione salva anche il valore restituito nell'oggetto puntato da questo.	26.3

tmpfile <i>Creazione di un file temporaneo</i>	<stdio.h>
<code>FILE *tmpfile(void);</code>	
Crea un file temporaneo che verrà rimosso automaticamente quando questo viene chiuso o il programma termina. Apre il file in modalità "wb+".	
<i>Restituisce</i>	
Un puntatore a file che può essere usato per eseguire successivamente delle operazioni sul file. Restituisce un puntatore nullo se il file temporaneo non può essere creato.	22.2

tmpnam <i>Generazione di un nome file temporaneo</i>	<stdio.h>
<code>char *tmpnam(char *s);</code>	
Genera un nome per un file temporaneo. Se s è un puntatore nullo, la funzione salva il nome del file in un oggetto statico. Negli altri casi copia il nome nel vettore di caratteri puntato da s (il vettore deve essere abbastanza lungo da salvare <code>L_tmpnam</code> caratteri).	
<i>Restituisce</i>	
Un puntatore al nome di un file. Se il nome non può essere generato restituisce un puntatore nullo.	22.2

tolower <i>Conversione a lettera minuscola</i>	<ctype.h>
<code>int tolower(int c);</code>	
<i>Restituisce</i>	
Se c è una lettera maiuscola la funzione restituisce la lettera minuscola corrispondente. Se c non è una lettera maiuscola, il carattere c viene restituito senza essere modificato.	23.5

toupper <i>Conversione a lettera maiuscola</i>	<ctype.h>
<code>int toupper(int c);</code>	

Restituisce

Se `c` è una lettera minuscola la funzione restituisce la lettera maiuscola corrispondente. Se `c` non è una lettera minuscola, il carattere `c` viene restituito senza essere modificato. 23.5

towctrans *Translitterazione di wide character (C99)*

<wctype.h>

```
wint_t towctrans(wint_t wc, wctrans_t desc);
```

Restituisce

Il valore mappato di `wc` utilizzando la mappatura descritta da `desc` (`desc` deve essere un valore restituito da una chiamata alla `wctrans` e l'impostazione corrente della categoria `LC_CTYPE` deve essere la stessa durante entrambe le chiamate). 25.6

towlower *Conversione in lettera minuscola di un wide character (C99)* <wctype.h>

```
wint_t tolower(wint_t wc);
```

Restituisce

Se la chiamata `iswupper(wc)` è vera, restituisce il corrispondente wide character per il quale nella localizzazione corrente la `iswlower` è vera, ammesso che tale carattere esista. Negli altri casi restituisce il valore di `wc` senza modificarlo. 25.6

towupper *Conversione in lettera maiuscola di un wide character (C99)* <wctype.h>

```
wint_t towupper(wint_t wc);
```

Restituisce

Se la chiamata `iswlower(wc)` è vera, restituisce il corrispondente wide character per il quale nella localizzazione corrente la `iswupper` è vera, ammesso che tale carattere esista. Negli altri casi restituisce il valore di `wc` senza modificarlo. 25.6

trunc *Troncamento al valore interegrale più vicino (C99)*

<math.h>

```
double trunc(double x);
```

truncf

```
float truncf(float x);
```

truncl

```
long double truncl(long double x);
```

Restituisce

`x` arrotondato all'intero più vicino (nel formato a virgola mobile) che non abbia valore assoluto più grande. 23.4

ungetc *Rispristino di un carattere dalla lettura*

<stdio.h>

```
int ungetc(int c, FILE *stream);
```

Rimette il carattere `c` nello stream puntato da `stream` e azzerava l'indicatore di end-of-file associate allo stream stesso. Il numero di caratteri che può essere rimesso a posto da delle chiamate consecutive alla `ungetc` varia, solamente la prima chiamata ha garanzia di avere successo. Chiamare una funzione di posizionamento (`fseek`, `fsetpos` o `rewind`) fa sì che il carattere ripristinato venga perso.

Restituisce

`c` (il carattere rimesso a posto). Restituisce EOF se si tenta di rimettere a posto il carattere EOF o se si cerca di rimettere a posto troppi caratteri senza un'operazione di lettura o di posizionamento nel file. 22.4

ungetwc *Rispristino di un wide character dalla lettura (C99)*

<wchar.h>

```
wint_t ungetwc(wint_t c, FILE *stream);
```

Versione wide character della `ungetc`.

25.5**va_arg** *Caricamento di un argomento dall'elenco variabile di argomenti* <stdarg.h>

```
type va_arg(va_list ap, type);
```

macro

Carica un argomento nell'elenco variabile di argomenti associato con `ap` e poi modifica lo stesso `ap` in modo che il prossimo utilizzo della funzione carichi l'argomento seguente. Il parametro `ap` deve essere inizializzato dalla funzione `va_start` (o dalla funzione `va_copy` nel C99) precedentemente alla prima chiamata alla funzione.

Restituisce

Il valore dell'argomento assumendo che il suo tipo (dopo l'applicazione delle promozioni di default degli argomenti) sia compatibile con `type`. 26.1

va_copy *Copia dell'elenco variabile di argomenti (C99)*

<stdarg.h>

```
void va_copy(va_list dest, va_list src);
```

macro

Copia `src` dentro `dest`. Il valore di `dest` diverrà lo stesso che si avrebbe avuto se ad esso fosse stata applicata la

funzione `va_start` seguita dalla stessa sequenza di applicazioni della funzione `va_arg` che è stata utilizzata per raggiungere l'attuale stato di `src`. 26.1

va_end *Fine dell'elaborazione dell'elenco variabile di argomenti* <stdarg.h>
 void va_end(va_list ap); macro

Termina l'elaborazione dell'elenco variabile di argomenti associate ad `ap`. 26.1

va_start *Inizio dell'elaborazione dell'elenco variabile di argomenti* <stdarg.h>
 void va_start(va_list ap, parmN); macro

Deve essere invocata prima di accedere all'elenco variabile di argomenti. Inizializza `ap` per il successivo utilizzo con le funzioni `va_arg` e `va_end`. `parmN` è il nome dell'ultimo parametro ordinario (quello seguito da `,` `...`). 26.1

vfprintf *Scrittura formattata su file usando un elenco variabile di argomenti* <stdio.h>
 int vfprintf(FILE * restrict stream,
 const char * restrict format,
 va_list arg);

Equivalente alla `fprintf` con l'elenco variabile di argomenti sostituito da `arg`.

Restituisce

Il numero di caratteri scritti. Se si verifica un errore restituisce un valore negativo. 26.1

vfscanf *Letture formattata su file usando un elenco variabile di argomenti (C99)* <stdio.h>
 int vfscanf(FILE * restrict stream,
 const char * restrict format,
 va_list arg);

Equivalente alla `fscanf` con l'elenco variabile di argomenti sostituito da `arg`.

Restituisce

Il numero di oggetti di input letti e salvati con successo. Restituisce EOF se si verifica un errore nell'input prima che qualsiasi oggetto possa essere letto. 26.1

vfwprintf *Scrittura formattata di wide character su file usando un elenco variabile di argomenti (C99)* <wchar.h>
 int vfwprintf(FILE * restrict stream,
 const wchar_t * restrict format,
 va_list arg);

Versione wide character della `vfprintf`. 25.5

fwscanf *Letture formattata di wide character su file usando un elenco variabile di argomenti (C99)* <wchar.h>
 int fwscanf(FILE * restrict stream,
 const wchar_t * restrict format,
 va_list arg);

Versione wide character della `vfscanf`. 25.5

vprintf *Scrittura formattata usando un elenco variabile di argomenti* <stdio.h>
 int vprintf(const char * restrict format, va_list arg);

Equivalente alla `printf` con l'elenco variabile di argomenti sostituito da `arg`.

Restituisce

Il numero di caratteri scritti. Se si verifica un errore restituisce un valore negativo. 26.1

vscanf *Letture formattata usando un elenco variabile di argomenti (C99)* <stdio.h>
 int vscanf(const char * restrict format, va_list arg);

Equivalente alla `scanf` con l'elenco variabile di argomenti sostituito da `arg`.

Restituisce

Il numero di oggetti di input letti e salvati con successo. Restituisce EOF se si verifica un errore nell'input prima che qualsiasi oggetto possa essere letto. 26.1

vsnprintf *Scrittura formattata su stringa con limite usando* <stdio.h>

un elenco variabile di argomenti (C99)

```
int vsnprintf(char * restrict s, size_t n,
             const char * restrict format,
             va_list arg);
```

Equivalente alla `snprintf` con l'elenco variabile di argomenti sostituito da `arg`.

Restituisce

Il numero di caratteri che sarebbero stati salvati nel vettore puntato da `s` (senza includere il carattere null) se non ci fosse stata alcuna restrizione sulla lunghezza. Se si verifica un errore di codifica restituisce un valore negativo. 26.1

vsprintf *Scrittura formattata su stringa usando un elenco variabile* <stdio.h>

di argomenti

```
int vsprintf(char * restrict s,
            const char * restrict format,
            va_list arg);
```

Equivalente alla `sprintf` con l'elenco variabile di argomenti sostituito da `arg`.

Restituisce

Il numero di caratteri salvati nel vettore puntato da `s`, senza includere il carattere null. Nel C99 restituisce un valore negativo nel caso si verificasse un errore di codifica. 26.1

vsscanf *Scrittura formattata su stringa usando un elenco variabile* <stdio.h>

di argomenti (C99)

```
int vsscanf(const char * restrict s,
           const char * restrict format,
           va_list arg);
```

Equivalente alla `sscanf` con l'elenco variabile di argomenti sostituito da `arg`.

Restituisce

Il numero di oggetti di input letti e salvati con successo. Restituisce EOF se si verifica un errore nella lettura prima che gli oggetti possano essere letti. 26.1

vswprintf *Scrittura formattata wide character su stringa usando* <wchar.h>

un elenco variabile di argomenti (C99)

```
int vswprintf(wchar_t * restrict s, size_t n,
             const wchar_t * restrict format,
             va_list arg);
```

Equivalente alla `swprintf` con l'elenco variabile di argomenti sostituito da `arg`.

Restituisce

Il numero di wide character salvati con successo nel vettore puntato da `s`, senza includere il wide character null. Restituisce un valore negativo se si verifica un errore di codifica o se il numero di wide character che deve essere scritto è pari a `n` o più. 25.5

vswscanf *Scrittura formattata wide character su stringa usando un* <wchar.h>

elenco variabile di argomenti (C99)

```
int vswscanf(const wchar_t * restrict s,
            const wchar_t * restrict format,
            va_list arg);
```

Versione wide character della `vsscanf`.

25.5

vwprintf *Scrittura formattata wide character usando un elenco* <wchar.h>

variabile di argomenti (C99)

```
int vwprintf(const wchar_t * restrict format,
            va_list arg);
```

Versione wide character della `vprintf`.

25.5

vwscanf *Lettura formattata wide character usando un elenco* <wchar.h>

variabile di argomenti (C99)

```
int vwscanf(const wchar_t * restrict format,
```

```
va_list arg);
```

 Versione wide character della `vscanf`.

25.5

wcrtomb *Conversione di un wide character in un carattere multibyte – Riavviabile (C99)* <wchar.h>

```
size_t wcrtomb(char * restrict s, wchar_t wc,
               mbstate_t * restrict ps);
```

Se `s` è un puntatore nullo una chiamata alla funzione è equivalente a

```
wcrtomb(buf, L'\0', ps)
```

dove `buf` è un buffer interno. Negli altri casi la funzione converte `wc` dalla forma wide character in quella di un carattere multibyte (incluso eventualmente delle sequenze di shift) il quale viene salvato nel vettore puntato da `s`. Il valore di `ps` deve essere un puntatore a un oggetto di tipo `mbstate_t` che contenga il corrente stato di conversione. Se `ps` è un puntatore nullo la funzione utilizza un oggetto interno per salvare lo stato di conversione. Se `wc` corrisponde al wide character null, la funzione salva un byte null preceduto da una sequenza di shift, nel caso fosse necessaria per ripristinare lo stato iniziale di shift, mentre l'oggetto `mbstate_t` utilizzato durante la chiamata viene lasciato nello stato di conversione iniziale.

Restituisce

Il numero di byte salvati nel vettore includendo le sequenze di shift. Se `wc` non è un wide character valido la funzione restituisce il valore (`size_t`) (-1) e salva il valore `EILSEQ` nella variabile `errno`. 25.5

wcscat *Concatenamento di stringhe wide (C99)* <wchar.h>

```
wchar_t *wcscat(wchar_t * restrict s1,
                const wchar_t * restrict s2);
```

 Versione wide character della `strcat`.

25.5

wcschr *Ricerca di un carattere in una stringa wide (C99)* <wchar.h>

```
wchar_t *wcschr(const wchar_t *s, wchar_t c);
```

 Versione wide character della `strchr`.

25.5

wcscmp *Confronto di stringhe wide (C99)* <wchar.h>

```
int wcscmp(const wchar_t *s1, const wchar_t *s2);
```

 Versione wide character della `strcmp`.

25.5

wcscoll *Confronto di stringhe wide usando una sequenza di confronto specifica per la localizzazione (C99)* <wchar.h>

```
int wcscoll(const wchar_t *s1, const wchar_t *s2);
```

 Versione wide character della `strcoll`.

25.5

wcscpy *Copia di stringa wide (C99)* <wchar.h>

```
wchar_t *wcscpy(wchar_t * restrict s1,
                const wchar_t * restrict s2);
```

 Versione wide character della `strcpy`.

25.5

wcscspn *Ricerca in una stringa wide di uno span iniziale non appartenente al set (C99)* <wchar.h>

```
size_t wcscspn(const wchar_t *s1, const wchar_t *s2);
```

 Versione wide character della `strcspn`.

25.5

wcsftime *Scrittura formattata di data e ora in una stringa wide (C99)* <wchar.h>

```
size_t wcsftime(wchar_t * restrict s, size_t maxsize,
                const wchar_t * restrict format,
                const struct tm * restrict timeptr);
```

 Versione wide character della `strftime`.

25.5

wcslen *Lunghezza di stringa wide (C99)* <wchar.h>

```
size_t wcslen(const wchar_t *s);
```

 Versione wide character della `strlen`.

25.5

wcsncat Concatenamento limitato di stringhe wide (C99)	<wchar.h>
<pre>wchar_t *wcsncat(wchar_t * restrict s1, const wchar_t * restrict s2, size_t n);</pre>	
Versione wide character della strncat.	25.5
wcsncmp Confronto di stringhe wide con limite (C99)	<wchar.h>
<pre>int wcsncmp(const wchar_t *s1, const wchar_t *s2, size_t n);</pre>	
Versione wide character della strncmp.	25.5
wcsncpy Copia limitata di stringa wide (C99)	<wchar.h>
<pre>wchar_t *wcsncpy(wchar_t * restrict s1, const wchar_t * restrict s2, size_t n);</pre>	
Versione wide character della strncpy.	25.5
wcspbrk Ricerca in una stringa wide di un carattere appartenente al set (C99)	<wchar.h>
<pre>wchar_t *wcspbrk(const wchar_t *s1, const wchar_t *s2);</pre>	
Versione wide character della strpbrk.	25.5
wcsrchr Ricerca in ordine inverso di un carattere all'interno di una stringa wide (C99)	<wchar.h>
<pre>wchar_t *wcsrchr(const wchar_t *s, wchar_t c);</pre>	
Versione wide character della strrchr.	25.5
wcsrtombs Conversione di una stringa wide in una stringa multibyte – Riavviabile (C99)	<wchar.h>
<pre>size_t wcsrtombs(char * restrict dst, const wchar_t ** restrict src, size_t len, mbstate_t * restrict ps);</pre>	
<p>Converte la sequenza di wide character presente nel vettore puntato indirettamente da <code>src</code> in una sequenza di corrispondenti caratteri multibyte che inizia con lo stato di conversione descritto dall'oggetto puntato da <code>ps</code>. Se <code>ps</code> è un puntatore nullo, la funzione utilizza un oggetto interno per salvare lo stato di conversione. Se <code>dst</code> non è un puntatore nullo, i caratteri convertiti vengono poi salvati nel vettore puntato dallo stesso <code>dst</code>. La conversione continua fino al carattere null di termine che viene a sua volta memorizzato. La conversione termina prima se viene incontrato un wide character che non corrisponda a un carattere multibyte valido o, nel caso <code>dst</code> non fosse un puntatore nullo, se il prossimo carattere multibyte eccederebbe il limite dei <code>len</code> byte totali che devono essere salvati nel vettore puntato dallo stesso <code>dst</code>. Se <code>dst</code> non è un puntatore nullo, all'oggetto puntato da <code>src</code> viene assegnato un puntatore nullo (se il wide character null di termine era stato raggiunto) oppure l'indirizzo immediatamente successivo all'ultimo wide character convertito (se esiste). Se la conversione finisce con un wide character null, lo stato di conversione risultante è quello iniziale.</p> <p><i>Restituisce</i></p> <p>Il numero di byte presenti nella sequenza risultante di caratteri multibyte senza includere qualsiasi carattere null di termine. Se viene incontrato un wide character non corrispondente a un carattere multibyte valido restituisce <code>(size_t) (-1)</code> e salva il valore <code>EILSEQ</code> nella variabile <code>errno</code>.</p>	
wcsspn Ricerca di uno span iniziale di caratteri appartenenti a un set all'interno di una stringa wide (C99)	<wchar.h>
<pre>size_t wcsspn(const wchar_t *s1, const wchar_t *s2);</pre>	
Versione wide character della strspn.	25.5
wcsstr Ricerca di una sottostringa in una stringa wide (C99)	<wchar.h>
<pre>wchar_t *wcsstr(const wchar_t *s1, const wchar_t *s2);</pre>	
Versione wide character della strstr.	25.5

wcstod <i>Conversione di una stringa wide in un double (C99)</i>	<wchar.h>
<pre>double wcstod(const wchar_t * restrict nptr, wchar_t ** restrict endptr);</pre>	
Versione wide character della strtod.	25.5
wcstof <i>Conversione di una stringa wide in un float (C99)</i>	<wchar.h>
<pre>float wcstof(const wchar_t * restrict nptr, wchar_t ** restrict endptr);</pre>	
Versione wide character della strtof.	25.5
wcstoimax <i>Conversione di una stringa wide in un intero della dimensione più grande (C99)</i>	<inttypes.h>
<pre>intmax_t wcstoimax(const wchar_t * restrict nptr, wchar_t ** restrict endptr, int base);</pre>	
Versione wide character della strtoumax.	27.2
wcstok <i>Ricerca di un token in una stringa wide (C99)</i>	<wchar.h>
<pre>wchar_t *wcstok(wchar_t * restrict s1, const wchar_t * restrict s2, wchar_t ** restrict ptr);</pre>	
Ricerca all'interno della stringa wide puntata da <i>s1</i> un "token" che consista di caratteri non presenti nella stringa wide puntata da <i>s2</i> . Se il token esiste, il carattere che lo segue viene tramutato nel wide character null. Se <i>s1</i> è un puntatore nullo, viene continuata la ricerca iniziata da una precedente chiamata alla funzione. La ricerca inizia immediatamente dopo il wide character null presente alla fine del token precedente. L'argomento <i>ptr</i> punta a un oggetto di tipo <code>wchar_t *</code> che la funzione modifica per tenere traccia dei suoi progressi. Se <i>s1</i> è un puntatore nullo, l'oggetto deve essere lo stesso usato da una chiamata precedente e determina dentro quale stringa wide debba essere condotta la ricerca e dove questa ultima debba iniziare.	
<i>Restituisce</i>	
Un puntatore al primo wide character del token. Se il token non viene trovato restituisce un puntatore nullo.	
	25.5
wcstol <i>Conversione di una stringa wide in un long int (C99)</i>	<wchar.h>
<pre>long int wcstol(const wchar_t * restrict nptr, wchar_t ** restrict endptr, int base);</pre>	
Versione wide character della strtol.	25.5
wcstold <i>Conversione di una stringa wide in un long double (C99)</i>	<wchar.h>
<pre>long double wcstold(const wchar_t * restrict nptr, wchar_t ** restrict endptr);</pre>	
Versione wide character della strtold.	25.5
wcstoll <i>Conversione di una stringa wide in un long long int (C99)</i>	<wchar.h>
<pre>long long int wcstoll(const wchar_t * restrict nptr, wchar_t ** restrict endptr, int base);</pre>	
Versione wide character della strtoll.	25.5
wcstombs <i>Conversione di una stringa wide in una stringa multibyte</i>	<stdlib.h>
<pre>size_t wcstombs(char * restrict s, const wchar_t * restrict pwcs, size_t n);</pre>	
Converte una sequenza di wide character nei corrispondenti caratteri multibyte. L'argomento <i>pwcs</i> punta a un vettore contenente i wide character. I caratteri multibyte vengono salvati nel vettore puntato da <i>s</i> . La conversione termina se viene salvato il carattere null o se il salvataggio di un carattere multibyte eccede il limite di <i>n</i> byte.	
<i>Restituisce</i>	
Il numero di byte salvati senza includere il carattere null posto alla fine (se presente). Se viene incontrato un wide character che non corrisponde a un carattere multibyte valido restituisce <code>(size_t) (-1)</code> .	
	25.2

<p>wcstoul <i>Conversione di una stringa wide in un unsigned long int (C99)</i> <wchar.h></p> <pre>unsigned long int wcstoul(const wchar_t * restrict nptr, wchar_t ** restrict endptr, int base);</pre>	25.5
<hr/>	
<p>wcstoull <i>Conversione di una stringa wide in un unsigned long long int (C99)</i><wchar.h></p> <pre>unsigned long long int wcstoull(const wchar_t * restrict nptr, wchar_t ** restrict endptr, int base);</pre>	25.5
<hr/>	
<p>wcstoumax <i>Conversione di una stringa wide in un intero senza segno <inttypes.h> della dimensione più grande (C99)</i></p> <pre>uintmax_t wcstoumax(const wchar_t * restrict nptr, wchar_t ** restrict endptr, int base);</pre>	27.2
<hr/>	
<p>wcsxfrm <i>Trasformazione di una stringa wide (C99)</i> <wchar.h></p> <pre>size_t wcsxfrm(wchar_t * restrict s1, const wchar_t * restrict s2, size_t n);</pre>	25.5
<hr/>	
<p>wctob <i>Conversione un wide character in un byte (C99)</i> <wchar.h></p> <pre>int wctob(wint_t c);</pre> <p><i>Restituisce</i> Rappresentazione a singolo byte di c come un unsigned char convertito al tipo int. Se c non corrisponde a un carattere multibyte nello stato di shift iniziale restituisce EOF.</p>	25.5
<hr/>	
<p>wctomb <i>Conversione un wide character in un carattere multibyte</i> <stdlib.h></p> <pre>int wctomb(char *s, wchar_t wc);</pre> <p>Converte il wide character contenuto in wc in un carattere multibyte. Se s non è un puntatore nullo salva il risultato nel vettore puntato da s.</p> <p style="text-align: right;"><i>Restituisce</i> Se s è un puntatore nullo restituisce un valore diverso da zero o uguale a zero a seconda che i caratteri multibyte abbiano o meno delle codifiche dipendenti dallo stato. Negli altri casi restituisce il numero di byte presenti nel carattere multibyte che corrisponde a wc. Se wc non corrisponde a un carattere multibyte valido restituisce -1.</p>	25.2
<hr/>	
<p>wctrans <i>Definizione di una mappatura wide character (C99)</i> <wctype.h></p> <pre>wctrans_t wctrans(const char *property);</pre> <p><i>Restituisce</i> Se property identifica una mappatura valida di wide character in accordo con la categoria LC_CTYPE della localizzazione corrente, la funzione restituisce un valore diverso da zero che può essere usato come secondo argomento della funzione towctrans. Negli altri casi restituisce il valore zero.</p>	25.6
<hr/>	
<p>wctype <i>Definizione di una classe wide character (C99)</i> <wctype.h></p> <pre>wctype_t wctype(const char *property);</pre> <p><i>Restituisce</i> Se property identifica una mappatura valida di wide character in accordo con la categoria LC_CTYPE della localizzazione corrente, la funzione restituisce un valore diverso da zero che può essere usato come secondo argomento della funzione iswctype. otherwise, Negli altri casi restituisce il valore zero.</p>	25.6
<hr/>	
<p>wmemchr <i>Ricerca di un carattere in un blocco di memoria wide character (C99)</i><wchar.h></p> <pre>wchar_t *wmemchr(const wchar_t *s, wchar_t c, size_t n);</pre>	25.5

wmemcmp <i>Confronto di blocchi di memoria wide character (C99)</i>	<wchar.h>
<i>int wmemcmp(const wchar_t * s1, const wchar_t * s2, size_t n);</i>	
Versione wide character della memcmp.	25.5

wmemcpy <i>Copia di un blocco di memoria wide character (C99)</i>	<wchar.h>
<i>wchar_t *wmemcpy(wchar_t * restrict s1, const wchar_t * restrict s2, size_t n);</i>	
Versione wide character della memcpy.	25.5

wmemmove <i>Copia di un blocco di memoria wide character (C99)</i>	<wchar.h>
<i>wchar_t *wmemmove(wchar_t *s1, const wchar_t *s2, size_t n);</i>	
Versione wide character della memmove.	25.5

wmemset <i>Inizializzazione di un blocco di memoria wide character (C99)</i>	<wchar.h>
<i>wchar_t *wmemset(wchar_t *s, wchar_t c, size_t n);</i>	
Versione wide character della memset.	25.5

wprintf <i>Scrittura formattata wide character (C99)</i>	<wchar.h>
<i>int wprintf(const wchar_t * restrict format, ...);</i>	
Versione wide character della printf.	25.5

wscanf <i>Lettura formattata wide character (C99)</i>	<wchar.h>
<i>int wscanf(const wchar_t * restrict format, ...);</i>	
Versione wide character della scanf.	25.5