Next: Errors in Math Functions, Previous: Hyperbolic Functions, Up: Mathematics [Contents][Index]

## 19.6 Special Functions

These are some more exotic mathematical functions which are sometimes useful. Currently they only have real-valued versions.

Function: *double* **erf** *(double x)*
Function: *float* **erff** *(float x)*
Function: *long double* **erfl** *(long double x)*
Function: *_FloatN* **erffN** *(_FloatN x)*
Function: *_FloatNx* **erffNx** *(_FloatNx x)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

`erf` returns the error function of *x*. The error function is defined as

```
erf (x) = 2/sqrt(pi) * integral from 0 to x of exp(-t^2) dt
```

Function: *double* **erfc** *(double x)*
Function: *float* **erfcf** *(float x)*
Function: *long double* **erfcl** *(long double x)*
Function: *_FloatN* **erfcfN** *(_FloatN x)*
Function: *_FloatNx* **erfcfNx** *(_FloatNx x)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

`erfc` returns `1.0 - erf(x)`, but computed in a fashion that avoids round-off error when *x* is large.

Function: *double* **lgamma** *(double x)*
Function: *float* **lgammaf** *(float x)*
Function: *long double* **lgammal** *(long double x)*
Function: *_FloatN* **lgammafN** *(_FloatN x)*
Function: *_FloatNx* **lgammafNx** *(_FloatNx x)*

Preliminary: | MT-Unsafe race:signgam | AS-Unsafe | AC-Safe | See POSIX Safety Concepts.

`lgamma` returns the natural logarithm of the absolute value of the gamma function of *x*. The gamma function is defined as

```
gamma (x) = integral from 0 to &infin; of t^(x-1) e^-t dt
```

The sign of the gamma function is stored in the global variable *signgam*, which is declared in `math.h`. It is `1` if the intermediate result was positive or zero, or `-1` if it was negative.

To compute the real gamma function you can use the `tgamma` function or you can compute the values as follows:

```
lgam = lgamma(x);
gam  = signgam*exp(lgam);
```

The gamma function has singularities at the non-positive integers. `lgamma` will raise the zero divide exception if evaluated at a singularity.

Function: *double* **lgamma_r** *(double x, int *signp)*
Function: *float* **lgammaf_r** *(float x, int *signp)*
Function: *long double* **lgammal_r** *(long double x, int *signp)*
Function: *_FloatN* **lgammafN_r** *(_FloatN x, int *signp)*
Function: *_FloatNx* **lgammafNx_r** *(_FloatNx x, int *signp)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

`lgamma_r` is just like `lgamma`, but it stores the sign of the intermediate result in the variable pointed to by *signp* instead of in the *signgam* global. This means it is reentrant.

The `lgammafN_r` and `lgammafNx_r` functions are GNU extensions.

Function: *double* **gamma** *(double x)*
Function: *float* **gammaf** *(float x)*
Function: *long double* **gammal** *(long double x)*

Preliminary: | MT-Unsafe race:signgam | AS-Unsafe | AC-Safe | See POSIX Safety Concepts.

These functions exist for compatibility reasons. They are equivalent to `lgamma` etc. It is better to use `lgamma` since for one the name reflects better the actual computation, and moreover `lgamma` is standardized in ISO C99 while `gamma` is not.

Function: *double* **tgamma** *(double x)*
Function: *float* **tgammaf** *(float x)*
Function: *long double* **tgammal** *(long double x)*
Function: *_FloatN* **tgammafN** *(_FloatN x)*
Function: *_FloatNx* **tgammafNx** *(_FloatNx x)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

`tgamma` applies the gamma function to *x*. The gamma function is defined as

```
gamma (x) = integral from 0 to &infin; of t^(x-1) e^-t dt
```

This function was introduced in ISO C99. The `_FloatN` and `_FloatNx` variants were introduced in ISO/IEC TS 18661-3.

Function: *double* **j0** *(double x)*

Function: *float* **j0f** *(float x)*
Function: *long double* **j0l** *(long double x)*
Function: *_FloatN* **j0fN** *(_FloatN x)*
Function: *_FloatNx* **j0fNx** *(_FloatNx x)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

`j0` returns the Bessel function of the first kind of order 0 of *x*. It may signal underflow if *x* is too large.

The `_FloatN` and `_FloatNx` variants are GNU extensions.

Function: *double* **j1** *(double x)*
Function: *float* **j1f** *(float x)*
Function: *long double* **j1l** *(long double x)*
Function: *_FloatN* **j1fN** *(_FloatN x)*
Function: *_FloatNx* **j1fNx** *(_FloatNx x)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

`j1` returns the Bessel function of the first kind of order 1 of *x*. It may signal underflow if *x* is too large.

The `_FloatN` and `_FloatNx` variants are GNU extensions.

Function: *double* **jn** *(int n, double x)*
Function: *float* **jnf** *(int n, float x)*
Function: *long double* **jnl** *(int n, long double x)*
Function: *_FloatN* **jnfN** *(int n, _FloatN x)*
Function: *_FloatNx* **jnfNx** *(int n, _FloatNx x)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

`jn` returns the Bessel function of the first kind of order *n* of *x*. It may signal underflow if *x* is too large.

The `_FloatN` and `_FloatNx` variants are GNU extensions.

Function: *double* **y0** *(double x)*
Function: *float* **y0f** *(float x)*
Function: *long double* **y0l** *(long double x)*
Function: *_FloatN* **y0fN** *(_FloatN x)*
Function: *_FloatNx* **y0fNx** *(_FloatNx x)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

`y0` returns the Bessel function of the second kind of order 0 of *x*. It may signal underflow if *x* is too large. If *x* is negative, `y0` signals a domain error; if it is zero, `y0` signals overflow and returns -*∞*.

The `_FloatN` and `_FloatNx` variants are GNU extensions.

Function: *double* **y1** *(double x)*
Function: *float* **y1f** *(float x)*
Function: *long double* **y1l** *(long double x)*
Function: *_FloatN* **y1fN** *(_FloatN x)*
Function: *_FloatNx* **y1fNx** *(_FloatNx x)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

`y1` returns the Bessel function of the second kind of order 1 of *x*. It may signal underflow if *x* is too large. If *x* is negative, `y1` signals a domain error; if it is zero, `y1` signals overflow and returns -*∞*.

The `_FloatN` and `_FloatNx` variants are GNU extensions.

Function: *double* **yn** *(int n, double x)*
Function: *float* **ynf** *(int n, float x)*
Function: *long double* **ynl** *(int n, long double x)*
Function: *_FloatN* **ynfN** *(int n, _FloatN x)*
Function: *_FloatNx* **ynfNx** *(int n, _FloatNx x)*

Preliminary: | MT-Safe | AS-Safe | AC-Safe | See POSIX Safety Concepts.

`yn` returns the Bessel function of the second kind of order *n* of *x*. It may signal underflow if *x* is too large. If *x* is negative, `yn` signals a domain error; if it is zero, `yn` signals overflow and returns -*∞*.

The `_FloatN` and `_FloatNx` variants are GNU extensions.

---

Next: Errors in Math Functions, Previous: Hyperbolic Functions, Up: Mathematics [Contents][Index]