Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'atan2.3p' command

$ man atan2.3p

ATAN2(3P) POSIX Programmer's Manual ATAN2(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

atan2, atan2f, atan2l ? arc tangent functions

SYNOPSIS

```
#include <math.h>
double atan2(double y, double x);
float atan2f(float y, float x);
long double atan2l(long double y, long double x);
```

DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1?2017 defers to the ISO C standard.

These functions shall compute the principal value of the arc tangent of y/x, using the signs of both arguments to determine the quadrant of the return value.

An application wishing to check for error situations should set errno to zero and call feclearexcept(FE_ALL_EXCEPT) before calling these

functions. On return, if errno is non-zero or fetestexcept(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW) is non-zero, an error has occurred.

RETURN VALUE

Upon successful completion, these functions shall return the arc tan? gent of y/x in the range [-?,?] radians.

If y is ?0 and x is < 0, ?? shall be returned.

If y is ?0 and x is > 0, ?0 shall be returned.

If y is < 0 and x is ?0, -?/2 shall be returned.

If y is > 0 and x is ?0, ?/2 shall be returned.

If x is 0, a pole error shall not occur.

If either x or y is NaN, a NaN shall be returned.

If the correct value would cause underflow, a range error may occur, and atan(), atan2f(), and atan2l() shall return an implementation-de? fined value no greater in magnitude than DBL_MIN, FLT_MIN, and LDBL_MIN, respectively.

If the IEC 60559 Floating-Point option is supported, y/x should be re? turned.

If y is ?0 and x is -0, ?? shall be returned.

If y is ?0 and x is +0, ?0 shall be returned.

For finite values of ?y > 0, if x is -Inf, ?? shall be returned.

For finite values of ?y > 0, if x is +Inf, ?0 shall be returned.

For finite values of x, if y is ?Inf, ??/2 shall be returned.

If y is ?Inf and x is -Inf, ?3?/4 shall be returned.

If y is ?Inf and x is +Inf, ??/4 shall be returned.

If both arguments are 0, a domain error shall not occur.

ERRORS

These functions may fail if:

Range Error The result underflows.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then errno shall be set to [ERANGE]. If the integer expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the underflow floating-point exception shall

be raised.

The following sections are informative.

EXAMPLES

Converting Cartesian to Polar Coordinates System

The function below uses atan2() to convert a 2d vector expressed in cartesian coordinates (x,y) to the polar coordinates (rho,theta).

There are other ways to compute the angle theta, using asin() acos(), or atan(). However, atan2() presents here two advantages:

* The angle's quadrant is automatically determined.
* The singular cases (0,y) are taken into account.

Finally, this example uses hypot() rather than sqrt() since it is bet?ter for special cases; see hypot() for more information.

```
#include <math.h>
void
cartesian_to_polar(const double x, const double y,
                   double *rho, double *theta
    )
{
    *rho   = hypot (x,y); /* better than sqrt(x*x+y*y) */
    *theta = atan2 (y,x);
}
```

APPLICATION USAGE

On error, the expressions (math_errhandling & MATH_ERRNO) and (math_er?rhandling & MATH_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

acos(), asin(), atan(), feclearexcept(), fetestexcept(), hypot(), is?nan(), sqrt(), tan()

The Base Definitions volume of POSIX.1?2017, Section 4.20, Treatment of

Error Conditions for Mathematical Functions, <math.h>

COPYRIGHT