NAME

matherr – math library exception-handling function

SYNOPSIS

#include <math.h>

int matherr(struct exception *∗exc***);**

MT-LEVEL

MT-Safe

DESCRIPTION

The SVID3 (*System V Interface Definition Third Edition*) specifies that certain **libm** functions call **matherr( )** when exceptions are detected. Users may define their own mechanisms for handling exceptions, by including a function named **matherr( )** in their programs. **matherr( )** is of the form described above. When an exception occurs, a pointer to the exception structure *exc* will be passed to the user-supplied **matherr( )** function. This structure, which is defined in the **<math.h>** header file, is as follows:

```
struct exception {
        int type;
        char *name;
        double arg1, arg2, retval;
};
```

The element **type** is an integer describing the type of exception that has occurred, from the following list of constants (defined in the header file):

DOMAIN	argument domain exception
SING	argument singularity
OVERFLOW	overflow range exception
UNDERFLOW	underflow range exception
TLOSS	total loss of significance
PLOSS	partial loss of significance

Note that both **TLOSS** and **PLOSS** reflect limitations of particular algorithms for trigonometric functions that suffer abrupt declines in accuracy at definite boundaries. Since the Sun implementation does not suffer such abrupt declines, **PLOSS** is never signaled. **TLOSS** is signaled for Bessel functions *only* to satisfy SVID3 requirements.

The element **name** points to a string containing the name of the function that incurred the exception. The elements **arg1** and **arg2** are the arguments with which the function was invoked. **retval** is set to the default value that will be returned by the function unless the user's **matherr( )** sets it to a different value.

If the user's **matherr( )** function returns non-zero, no exception message will be printed, and **errno** will not be set.

SVID3 STANDARD CONFORMANCE

If **matherr( )** is not supplied by the user, the default matherr exception-handling mechanisms, summarized in the table below, will be invoked upon exception:

DOMAIN

0.0 is usually returned, **errno** is set to **EDOM**, and a message is usually printed on standard error.

SING The largest finite single-precision number, **HUGE** of appropriate sign is returned, **errno** is set to **EDOM**, and a message is printed on standard error.

OVERFLOW
The largest finite single-precision number, **HUGE** of appropriate sign is usually returned, **errno** is set to **ERANGE**.

UNDERFLOW
0.0 is returned, and **errno** is set to **ERANGE**.

TLOSS 0.0 is returned, **errno** is set to **ERANGE**, and a message is printed on standard error.

In general, **errno** is not a reliable error indicator in that it may be unexpectedly set by a function in a handler for an asynchronous signal.

DEFAULT ERROR HANDLING PROCEDURES (SVID3)

	Types of Errors				
<math.h> type	DOMAIN	SING	OVERFLOW	UNDERFLOW	TLOSS
errno	EDOM	EDOM	ERANGE	ERANGE	ERANGE
IEEE Exception	Invalid Operation	Division by Zero	Overflow	Underflow	–
fp_exception_type	fp_invalid	fp_division	fp_overflow	fp_underflow	–
ACOS, ASIN (|x| > 1):	Md, 0.0	–	–	–	–
ACOSH (x < 1), ATANH (|x| > 1):	NaN	–	–	–	–
ATAN2 (0,0):	Md, 0.0	–	–	–	–
COSH, SINH:	–	–	±HUGE	–	–
EXP:	–	–	+HUGE	0.0	–
FMOD (x,0):	x	–	–	–	–
HYPOT:	–	–	+HUGE	–	–
J0, J1, JN (|x| > X_TLOSS):	–	–	–	–	Mt, 0.0
LGAMMA:					
usual cases	–	–	+HUGE	–	–
(x = 0 or –integer)	–	Ms, +HUGE	–	–	–
LOG, LOG10:					
(x < 0)	Md, –HUGE	–	–	–	–
(x = 0)	–	Ms, –HUGE	–	–	–
POW:					
usual cases	–	–	±HUGE	±0.0	–
(x < 0) ** (y not an integer)	Md, 0.0	–	–	–	–
0 ** 0	Md, 0.0	–	–	–	–
0 ** (y < 0)	Md, 0.0	–	–	–	–
REMAINDER (x,0):	NaN	–	–	–	–
SCALB:	–	–	±HUGE_VAL	±0.0	–
SQRT (x < 0):	Md, 0.0	–	–	–	–
Y0, Y1, YN:					
(x < 0)	Md, –HUGE	–	–	–	–
(x = 0)	–	Md, –HUGE	–	–	–
(x > X_TLOSS)	–	–	–	–	Mt, 0.0

ABBREVIATIONS	
Md	Message is printed (DOMAIN error).
Ms	Message is printed (SING error).
Mt	Message is printed (TLOSS error).
NaN	IEEE NaN result and invalid operation exception.
HUGE	Maximum finite single-precision floating-point number.
HUGE_VAL	IEEE ∞ result and division-by-zero exception.
X_TLOSS	The value X_TLOSS is defined in <values.h>.

The interaction of IEEE arithmetic and **matherr( )** is not defined when executing under IEEE rounding modes other than the default round to nearest: **matherr( )** may not be called on overflow or underflow, and the SUN-provided **matherr( )** may return results that differ from those in this table.

X/OPEN (XPG3) STANDARD CONFORMANCE

XPG3 (*X/Open Portability Guide Issue 3*) no longer sanctions the use of the **matherr( )** interface. The following table summarizes the values returned in the exceptional cases. In general, XPG3 dictates that as long as one of the input argument(s) is a NaN, NaN shall be returned. In particular, **pow(**NaN,0**)** = NaN.

X/Open (XPG3) ERROR HANDLING PROCEDURES (compile with "cc –Xa")

	Types of Errors				
<math.h> type	DOMAIN	SING	OVERFLOW	UNDERFLOW	TLOSS
errno	EDOM	EDOM	ERANGE	ERANGE	ERANGE
ACOS, ASIN (|x| > 1):	0.0	–	–	–	–
ATAN2 (0,0):	0.0	–	–	–	–
COSH, SINH:	–	–	{±HUGE_VAL}	–	–
EXP:	–	–	{+HUGE_VAL}	{0.0}	–
FMOD (x,0):	{NaN}	–	–	–	–
HYPOT:	–	–	{+HUGE_VAL}	–	–
J0, J1, JN (|x| > X_TLOSS):	–	–	–	–	{0.0}
LGAMMA:					
usual cases	–	–	{+HUGE_VAL}	–	–
(x = 0 or –integer)	–	+HUGE_VAL	–	–	–
LOG, LOG10:					
(x < 0)	–HUGE_VAL	–	–	–	–
(x = 0)	–	–HUGE_VAL	–	–	–
POW:					
usual cases	–	–	±HUGE_VAL	±0.0	–
(x < 0) ** (y not an integer)	0.0	–	–	–	–
0 ** 0	{1.0}	–	–	–	–
0 ** (y < 0)	{–HUGE_VAL}	–	–	–	–
SQRT (x < 0):	0.0	–	–	–	–
Y0, Y1, YN:					
(x < 0)	{–HUGE_VAL}	–	–	–	–
(x = 0)	–	{–HUGE_VAL}	–	–	–
(x > X_TLOSS)	–	–	–	–	0.0

ABBREVIATIONS	
{...}	**errno** is not to be relied upon in all braced cases.
NaN	IEEE NaN result and invalid operation exception.
HUGE_VAL	IEEE ∞ result and division-by-zero exception.
X_TLOSS	The value X_TLOSS is defined in <values.h>.

ANSI/ISO-C STANDARD CONFORMANCE

The ANSI/ISO-C standard covers a small subset of XPG3.

The following table summarizes the values returned in the exceptional cases.

ANSI/ISO-C ERROR HANDLING PROCEDURES (compile with "cc –Xc")				
	Types of Errors			
<math.h> type	DOMAIN	SING	OVERFLOW	UNDERFLOW
errno	EDOM	EDOM	ERANGE	ERANGE
ACOS, ASIN (| x | > 1):	0.0	–	–	–
ATAN2 (0,0):	0.0	–	–	–
EXP:	–	–	+HUGE_VAL	0.0
FMOD (x,0):	NaN	–	–	–
LOG, LOG10:				
(x < 0)	–HUGE_VAL	–	–	–
(x = 0)	–	–HUGE_VAL	–	–
POW:				
usual cases	–	–	±HUGE_VAL	±0.0
(x < 0) ** (y not an integer)	0.0	–	–	–
0 ** (y < 0)	–HUGE_VAL	–	–	–
SQRT (x < 0):	0.0	–	–	–

ABBREVIATIONS	
NaN	IEEE NaN result and invalid operation exception.
HUGE_VAL	IEEE ∞ result and division-by-zero exception.

EXAMPLES

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int
matherr(struct exception *x) {
        switch (x–>type) {
        case DOMAIN:
                /* change sqrt to return sqrt(–arg1), not NaN */
                if (!strcmp(x–>name, "sqrt")) {
                        x–>retval = sqrt(–x–>arg1);
                        return (0); /* print message and set errno */
                } /* FALLTHRU */
        case SING:
                /* all other domain or sing exceptions, print message and */
                /* abort */
                fprintf(stderr, "domain exception in %s\n", x–>name);
```

```
                abort( );
                break;
        }
        return (0); /* all other exceptions, execute default procedure */
}
```