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 'fma.3p' command

$ man fma.3p

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

fma, fmaf, fmal ? floating-point multiply-add

SYNOPSIS

```
#include <math.h>
double fma(double x, double y, double z);
float fmaf(float x, float y, float z);
long double fmal(long double x, long double y, long double z);
```

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 (x * y) + z, rounded as one ternary oper? ation: they shall compute the value (as if) to infinite precision and round once to the result format, according to the rounding mode charac? terized by the value of FLT_ROUNDS.

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 (x * y) + z, rounded as one ternary operation.

If the result overflows or underflows, a range error may occur. On systems that support the IEC 60559 Floating-Point option, if the result overflows a range error shall occur.

If x or y are NaN, a NaN shall be returned.

If x multiplied by y is an exact infinity and z is also an infinity but with the opposite sign, a domain error shall occur, and either a NaN (if supported), or an implementation-defined value shall be returned.

If one of x and y is infinite, the other is zero, and z is not a NaN, a domain error shall occur, and either a NaN (if supported), or an imple? mentation-defined value shall be returned.

If one of x and y is infinite, the other is zero, and z is a NaN, a NaN shall be returned and a domain error may occur.

If x*y is not 0*Inf nor Inf*0 and z is a NaN, a NaN shall be returned.

ERRORS

These functions shall fail if:

Domain Error

The value of x*y+z is invalid, or the value x*y is invalid and z is not a NaN.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then errno shall be set to [EDOM]. If the in? teger expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

Range Error The result overflows.

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 overflow floating-point exception shall be raised.

These functions may fail if:

Domain Error

The value x*y is invalid and z is a NaN.

If the integer expression (math_errhandling & MATH_ERRNO) is non-zero, then errno shall be set to [EDOM]. If the in? teger expression (math_errhandling & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

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.

Range Error The result overflows.

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 overflow floating-point exception shall be raised.

The following sections are informative.

EXAMPLES

None.

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

In many cases, clever use of floating (fused) multiply-add leads to much improved code; but its unexpected use by the compiler can under?

mine carefully written code. The FP_CONTRACT macro can be used to dis?
allow use of floating multiply-add; and the fma() function guarantees
its use where desired. Many current machines provide hardware floating
multiply-add instructions; software implementation can be used for oth?
ers.

FUTURE DIRECTIONS

None.

SEE ALSO

feclearexcept(), fetestexcept()

The Base Definitions volume of POSIX.1?2017, Section 4.20, Treatment of
Error Conditions for Mathematical Functions, <math.h>

COPYRIGHT