```
************************************************************
    5193 Tue Apr 30 23:28:34 2019
new/usr/src/head/fenv.h
10882 math headers should stop supporting K&R C
************************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
 */
/*
 * Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _FENV_H
#define _FENV_H

#include <sys/feature_tests.h>

#ifdef __cplusplus
extern "C" {
#endif

#ifndef __P
#ifdef __STDC__
#define __P(p)  p
#else
#define __P(p)  ()
#endif
#endif  /* !defined(__P) */

/*
 * Rounding modes
 */
#if defined(__sparc)

#define FE_TONEAREST    0
#define FE_TOWARDZERO   1
#define FE_UPWARD       2
#define FE_DOWNWARD     3

#elif defined(__i386) || defined(__amd64)

#define FE_TONEAREST    0
#define FE_DOWNWARD     1
#define FE_UPWARD       2
#define FE_TOWARDZERO   3
```

```
#endif

extern int fegetround(void);
extern int fesetround(int);
extern int fegetround __P((void));
extern int fesetround __P((int));

#if (defined(__i386) || defined(__amd64)) && \
        (!defined(_STRICT_STDC) || defined(__EXTENSIONS__))

#define FE_FLTPREC      0
#define FE_DBLPREC      2
#define FE_LDBLPREC     3

extern int fegetprec(void);
extern int fesetprec(int);
extern int fegetprec __P((void));
extern int fesetprec __P((int));

#endif

/*
 * Exception flags
 */
#if defined(__sparc)

#define FE_INEXACT      0x01
#define FE_DIVBYZERO    0x02
#define FE_UNDERFLOW    0x04
#define FE_OVERFLOW     0x08
#define FE_INVALID      0x10
#define FE_ALL_EXCEPT   0x1f

#elif defined(__i386) || defined(__amd64)

#define FE_INVALID      0x01
#define FE_DIVBYZERO    0x04
#define FE_OVERFLOW     0x08
#define FE_UNDERFLOW    0x10
#define FE_INEXACT      0x20
#define FE_ALL_EXCEPT   0x3d

#endif

typedef int fexcept_t;

extern int feclearexcept(int);
extern int feraiseexcept(int);
extern int fetestexcept(int);
extern int fegetexceptflag(fexcept_t *, int);
extern int fesetexceptflag(const fexcept_t *, int);
extern int feclearexcept __P((int));
extern int feraiseexcept __P((int));
extern int fetestexcept __P((int));
extern int fegetexceptflag __P((fexcept_t *, int));
extern int fesetexceptflag __P((const fexcept_t *, int));

#if !defined(_STRICT_STDC) || defined(__EXTENSIONS__)

/*
 * Exception handling extensions
 */
#define FEX_NOHANDLER   -1
#define FEX_NONSTOP     0
#define FEX_ABORT       1
```

```
#define FEX_SIGNAL      2
#define FEX_CUSTOM      3

#define FEX_INEXACT     0x001
#define FEX_DIVBYZERO   0x002
#define FEX_UNDERFLOW   0x004
#define FEX_OVERFLOW    0x008
#define FEX_INV_ZDZ     0x010
#define FEX_INV_IDI     0x020
#define FEX_INV_ISI     0x040
#define FEX_INV_ZMI     0x080
#define FEX_INV_SQRT    0x100
#define FEX_INV_SNAN    0x200
#define FEX_INV_INT     0x400
#define FEX_INV_CMP     0x800
#define FEX_INVALID     0xff0
#define FEX_COMMON      (FEX_INVALID | FEX_DIVBYZERO | FEX_OVERFLOW)
#define FEX_ALL         (FEX_COMMON | FEX_UNDERFLOW | FEX_INEXACT)
#define FEX_NONE        0

#define FEX_NUM_EXC     12

/* structure to hold a numeric value in any format used by the FPU */
typedef struct {
        enum fex_nt {
                fex_nodata      = 0,
                fex_int         = 1,
                fex_llong       = 2,
                fex_float       = 3,
                fex_double      = 4,
                fex_ldouble     = 5
        } type;
        union {
                int             i;
#if !defined(_STRICT_STDC) && !defined(_NO_LONGLONG) || defined(_STDC_C99) || \
        defined(__C99FEATURES__)
                long long       l;
#else
                struct {
                        int     l[2];
                } l;
#endif
                float           f;
                double          d;
                long double     q;
        } val;
} fex_numeric_t;
```

______***unchanged_portion_omitted_***

```
extern int fex_get_handling(int);
extern int fex_set_handling(int, int, void (*)());
extern int fex_get_handling __P((int));
extern int fex_set_handling __P((int, int, void (*)()));

extern void fex_getexcepthandler(fex_handler_t *, int);
extern void fex_setexcepthandler(const fex_handler_t *, int);
extern void fex_getexcepthandler __P((fex_handler_t *, int));
extern void fex_setexcepthandler __P((const fex_handler_t *, int));

#ifdef __STDC__
#include <stdio_tag.h>
#ifndef _FILEDEFED
#define _FILEDEFED
typedef __FILE FILE;
#endif
#endif
```

```
extern FILE *fex_get_log(void);
extern int fex_set_log(FILE *);
extern int fex_get_log_depth(void);
extern int fex_set_log_depth(int);
extern void fex_log_entry(const char *);
extern FILE *fex_get_log __P((void));
extern int fex_set_log __P((FILE *));
extern int fex_get_log_depth __P((void));
extern int fex_set_log_depth __P((int));
extern void fex_log_entry __P((const char *));

#define __fex_handler_t fex_handler_t

#else

typedef struct {
        int     __mode;
        void    (*__handler)();
} __fex_handler_t[12];
```

______***unchanged_portion_omitted_***

```
#ifdef __STDC__
extern const fenv_t __fenv_dfl_env;
#else
extern fenv_t __fenv_dfl_env;
#endif

#define FE_DFL_ENV      (&__fenv_dfl_env)

extern int fegetenv(fenv_t *);
extern int fesetenv(const fenv_t *);
extern int feholdexcept(fenv_t *);
extern int feupdateenv(const fenv_t *);
extern int fegetenv __P((fenv_t *));
extern int fesetenv __P((const fenv_t *));
extern int feholdexcept __P((fenv_t *));
extern int feupdateenv __P((const fenv_t *));

#if !defined(_STRICT_STDC) || defined(__EXTENSIONS__)
extern void fex_merge_flags(const fenv_t *);
extern void fex_merge_flags __P((const fenv_t *));
#endif

#ifdef __cplusplus
}
```

______***unchanged_portion_omitted_***

```
**********************************************************
    6482 Tue Apr 30 23:28:34 2019
new/usr/src/head/floatingpoint.h
10882 math headers should stop supporting K&R C
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*      Copyright (C) 1989 AT&T */
/*        All Rights Reserved */

/*
 * Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
 */
/*
 * Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _FLOATINGPOINT_H
#define _FLOATINGPOINT_H

#ifdef __STDC__
#include <stdio_tag.h>
#endif
#include <sys/ieeefp.h>

#ifdef __cplusplus
extern "C" {
#endif

/*
 * <floatingpoint.h> contains definitions for constants, types, variables,
 * and functions for:
 *      IEEE floating-point arithmetic base conversion;
 *      IEEE floating-point arithmetic modes;
 *      IEEE floating-point arithmetic exception handling.
 */

#ifndef __P
#ifdef __STDC__
#define __P(p)  p
#else
#define __P(p)  ()
#endif
#endif  /* !defined(__P) */

#if defined(__STDC__) && !defined(_FILEDEFED)
#define _FILEDEFED
```

```
typedef __FILE FILE;
#endif

typedef int sigfpe_code_type;   /* Type of SIGFPE code. */

typedef void (*sigfpe_handler_type)();  /* Pointer to exception handler */

#define SIGFPE_DEFAULT (void (*)())0    /* default exception handling */
#define SIGFPE_IGNORE  (void (*)())1    /* ignore this exception or code */
#define SIGFPE_ABORT   (void (*)())2    /* force abort on exception */

extern sigfpe_handler_type sigfpe(sigfpe_code_type, sigfpe_handler_type);
extern sigfpe_handler_type sigfpe __P((sigfpe_code_type, sigfpe_handler_type));

/*
 * Types for IEEE floating point.
 */
typedef float single;

#ifndef _EXTENDED
#define _EXTENDED
typedef unsigned extended[3];
#endif

typedef long double quadruple;  /* Quadruple-precision type. */

typedef unsigned fp_exception_field_type;
                                /*
                                 * A field containing fp_exceptions OR'ed
                                 * together.
                                 */
/*
 * Definitions for base conversion.
 */
#define DECIMAL_STRING_LENGTH 512       /* Size of buffer in decimal_record. */

typedef char decimal_string[DECIMAL_STRING_LENGTH];
                                /* Decimal significand. */

typedef struct {
        enum fp_class_type fpclass;
        int     sign;
        int     exponent;
        decimal_string ds;      /* Significand - each char contains an ascii */
                                /* digit, except the string-terminating */
                                /* ascii null. */
        int     more;           /* On conversion from decimal to binary, != 0 */
                                /* indicates more non-zero digits following */
                                /* ds. */
        int     ndigits;        /* On fixed_form conversion from binary to */
                                /* decimal, contains number of digits */
                                /* required for ds. */
} decimal_record;
______unchanged_portion_omitted_

extern void single_to_decimal(single *, decimal_mode *, decimal_record *,
    fp_exception_field_type *);
extern void double_to_decimal(double *, decimal_mode *, decimal_record *,
    fp_exception_field_type *);
extern void extended_to_decimal(extended *, decimal_mode *,
    decimal_record *, fp_exception_field_type *);
extern void quadruple_to_decimal(quadruple *, decimal_mode *,
    decimal_record *, fp_exception_field_type *);

extern void decimal_to_single(single *, decimal_mode *, decimal_record *,
    fp_exception_field_type *);
```

```
extern void decimal_to_double(double *, decimal_mode *, decimal_record *,
    fp_exception_field_type *);
extern void decimal_to_extended(extended *, decimal_mode *,
    decimal_record *, fp_exception_field_type *);
extern void decimal_to_quadruple(quadruple *, decimal_mode *,
    decimal_record *, fp_exception_field_type *);

extern void string_to_decimal(char **, int, int, decimal_record *,
    enum decimal_string_form *, char **);
extern void func_to_decimal(char **, int, int, decimal_record *,
extern void single_to_decimal __P((single *, decimal_mode *, decimal_record *,
                                fp_exception_field_type *));
extern void double_to_decimal __P((double *, decimal_mode *, decimal_record *,
                                fp_exception_field_type *));
extern void extended_to_decimal __P((extended *, decimal_mode *,
                                decimal_record *, fp_exception_field_type *));
extern void quadruple_to_decimal __P((quadruple *, decimal_mode *,
                                decimal_record *, fp_exception_field_type *));

extern void decimal_to_single __P((single *, decimal_mode *, decimal_record *,
                                fp_exception_field_type *));
extern void decimal_to_double __P((double *, decimal_mode *, decimal_record *,
                                fp_exception_field_type *));
extern void decimal_to_extended __P((extended *, decimal_mode *,
                                decimal_record *, fp_exception_field_type *));
extern void decimal_to_quadruple __P((quadruple *, decimal_mode *,
                                decimal_record *, fp_exception_field_type *));

extern void string_to_decimal __P((char **, int, int, decimal_record *,
                                enum decimal_string_form *, char **));
extern void func_to_decimal __P((char **, int, int, decimal_record *,
    enum decimal_string_form *, char **,
    int (*)(void), int *, int (*)(int));
extern void file_to_decimal(char **, int, int, decimal_record *,
                                int (*)(void), int *, int (*)(int)));
extern void file_to_decimal __P((char **, int, int, decimal_record *,
    enum decimal_string_form *, char **,
    FILE *, int *);
                                FILE *, int *));

extern char *seconvert(single *, int, int *, int *, char *);
extern char *sfconvert(single *, int, int *, int *, char *);
extern char *sgconvert(single *, int, int, char *);
extern char *econvert(double, int, int *, int *, char *);
extern char *fconvert(double, int, int *, int *, char *);
extern char *gconvert(double, int, int, char *);
extern char *qeconvert(quadruple *, int, int *, int *, char *);
extern char *qfconvert(quadruple *, int, int *, int *, char *);
extern char *qgconvert(quadruple *, int, int, char *);

extern char *ecvt(double, int, int *, int *);
extern char *fcvt(double, int, int *, int *);
extern char *gcvt(double, int, char *);
extern char *seconvert __P((single *, int, int *, int *, char *));
extern char *sfconvert __P((single *, int, int *, int *, char *));
extern char *sgconvert __P((single *, int, int, char *));
extern char *econvert __P((double, int, int *, int *, char *));
extern char *fconvert __P((double, int, int *, int *, char *));
extern char *gconvert __P((double, int, int, char *));
extern char *qeconvert __P((quadruple *, int, int *, int *, char *));
extern char *qfconvert __P((quadruple *, int, int *, int *, char *));
extern char *qgconvert __P((quadruple *, int, int, char *));

extern char *ecvt __P((double, int, int *, int *));
extern char *fcvt __P((double, int, int *, int *));
extern char *gcvt __P((double, int, char *));
```

```
#if __cplusplus >= 199711L
namespace std {
#endif
/*
 * ANSI C Standard says the following entry points should be
 * prototyped in <stdlib.h>.  They are now, but weren't before.
 */
extern double atof(const char *);
extern double strtod(const char *, char **);
extern double atof __P((const char *));
extern double strtod __P((const char *, char **));
#if __cplusplus >= 199711L
}
```
______***unchanged_portion_omitted_***

```
**********************************************************
   18816 Tue Apr 30 23:28:35 2019
new/usr/src/head/iso/math_c99.h
10882 math headers should stop supporting K&R C
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
 */
/*
 * Copyright 2005 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _ISO_MATH_C99_H
#define _ISO_MATH_C99_H

#include <sys/isa_defs.h>
#include <sys/feature_tests.h>

#ifdef __cplusplus
extern "C" {
#endif

#undef  FP_ZERO
#define FP_ZERO         0
#undef  FP_SUBNORMAL
#define FP_SUBNORMAL    1
#undef  FP_NORMAL
#define FP_NORMAL       2
#undef  FP_INFINITE
#define FP_INFINITE     3
#undef  FP_NAN
#define FP_NAN          4
#ifndef __P
#ifdef __STDC__
#define __P(p)  p
#else
#define __P(p)  ()
#endif
#endif  /* !defined(__P) */

#if defined(_STDC_C99) || _XOPEN_SOURCE - 0 >= 600 || defined(__C99FEATURES__)
#if defined(__GNUC__)
#undef  HUGE_VAL
#define HUGE_VAL        (__builtin_huge_val())
#undef  HUGE_VALF
```

```
#define HUGE_VALF       (__builtin_huge_valf())
#undef  HUGE_VALL
#define HUGE_VALL       (__builtin_huge_vall())
#undef  INFINITY
#define INFINITY        (__builtin_inff())
#undef  NAN
#define NAN             (__builtin_nanf(""))

/*
 * C99 7.12.3 classification macros
 */
#undef  isnan
#undef  isinf
#if __GNUC__ >= 4
#define isnan(x)        __builtin_isnan(x)
#define isinf(x)        __builtin_isinf(x)
#define fpclassify(x)   __builtin_fpclassify(FP_NAN, FP_INFINITE, FP_NORMAL, \
    FP_SUBNORMAL, FP_ZERO, x)
#define isfinite(x)     __builtin_isfinite(x)
#define isnormal(x)     __builtin_isnormal(x)
#define signbit(x)      __builtin_signbit(x)
#else  /* __GNUC__ >= 4 */
#else
#define isnan(x)        __extension__( \
                        { __typeof(x) __x_n = (x); \
                        __builtin_isunordered(__x_n, __x_n); })
#define isinf(x)        __extension__( \
                        { __typeof(x) __x_i = (x); \
                        __x_i == (__typeof(__x_i)) INFINITY || \
                        __x_i == (__typeof(__x_i)) (-INFINITY); })
#endif
#undef  isfinite
#define isfinite(x)     __extension__( \
                        { __typeof(x) __x_f = (x); \
                        !isnan(__x_f) && !isinf(__x_f); })
#undef  isnormal
#define isnormal(x)     __extension__( \
                        { __typeof(x) __x_r = (x); isfinite(__x_r) && \
                        (sizeof (__x_r) == sizeof (float) ? \
                        __builtin_fabsf(__x_r) >= __FLT_MIN__ : \
                        sizeof (__x_r) == sizeof (double) ? \
                        __builtin_fabs(__x_r) >= __DBL_MIN__ : \
                        __builtin_fabsl(__x_r) >= __LDBL_MIN__); })
#undef  fpclassify
#define fpclassify(x)   __extension__( \
                        { __typeof(x) __x_c = (x); \
                        isnan(__x_c) ? FP_NAN : \
                        isinf(__x_c) ? FP_INFINITE : \
                        isnormal(__x_c) ? FP_NORMAL : \
                        __x_c == (__typeof(__x_c)) 0 ? FP_ZERO : \
                        FP_SUBNORMAL; })
#undef  signbit
#if defined(_BIG_ENDIAN)
#define signbit(x)      __extension__( \
                        { __typeof(x) __x_s = (x); \
                        (int)(*(unsigned *)&__x_s >> 31); })
                        (int) (*(unsigned *) &__x_s >> 31); })
#elif defined(_LITTLE_ENDIAN)
#define signbit(x)      __extension__( \
                        { __typeof(x) __x_s = (x); \
                        (sizeof (__x_s) == sizeof (float) ? \
                        (int)(*(unsigned *)&__x_s >> 31) : \
                        (int) (*(unsigned *) &__x_s >> 31) : \
                        sizeof (__x_s) == sizeof (double) ? \
                        (int)(((unsigned *)&__x_s)[1] >> 31) : \
                        (int)(((unsigned short *)&__x_s)[4] >> 15)); })
```

```
#endif  /* defined(_BIG_ENDIAN) */
#endif  /* __GNUC__ >= 4 */
                        (int) (((unsigned *) &__x_s)[1] >> 31) : \
                        (int) (((unsigned short *) &__x_s)[4] >> 15)); })
#endif

/*
 * C99 7.12.14 comparison macros
 */
#undef  isgreater
#define isgreater(x, y)         __builtin_isgreater(x, y)
#undef  isgreaterequal
#define isgreaterequal(x, y)    __builtin_isgreaterequal(x, y)
#undef  isless
#define isless(x, y)            __builtin_isless(x, y)
#undef  islessequal
#define islessequal(x, y)       __builtin_islessequal(x, y)
#undef  islessgreater
#define islessgreater(x, y)     __builtin_islessgreater(x, y)
#undef  isunordered
#define isunordered(x, y)       __builtin_isunordered(x, y)
#else   /* defined(__GNUC__) */
#undef  HUGE_VAL
#define HUGE_VAL        __builtin_huge_val
#undef  HUGE_VALF
#define HUGE_VALF       __builtin_huge_valf
#undef  HUGE_VALL
#define HUGE_VALL       __builtin_huge_vall
#undef  INFINITY
#define INFINITY        __builtin_infinity
#undef  NAN
#define NAN             __builtin_nan

/*
 * C99 7.12.3 classification macros
 */
#undef  fpclassify
#define fpclassify(x)   __builtin_fpclassify(x)
#undef  isfinite
#define isfinite(x)     __builtin_isfinite(x)
#undef  isinf
#define isinf(x)        __builtin_isinf(x)
#undef  isnan
#define isnan(x)        __builtin_isnan(x)
#undef  isnormal
#define isnormal(x)     __builtin_isnormal(x)
#undef  signbit
#define signbit(x)      __builtin_signbit(x)

/*
 * C99 7.12.14 comparison macros
 */
#undef  isgreater
#define isgreater(x, y)         ((x) __builtin_isgreater(y))
#undef  isgreaterequal
#define isgreaterequal(x, y)    ((x) __builtin_isgreaterequal(y))
#undef  isless
#define isless(x, y)            ((x) __builtin_isless(y))
#undef  islessequal
#define islessequal(x, y)       ((x) __builtin_islessequal(y))
#undef  islessgreater
#define islessgreater(x, y)     ((x) __builtin_islessgreater(y))
#undef  isunordered
#define isunordered(x, y)       ((x) __builtin_isunordered(y))
#endif  /* defined(__GNUC__) */
#endif  /* defined(_STDC_C99) || _XOPEN_SOURCE - 0 >= 600 || ... */
```

```
#if defined(__EXTENSIONS__) || defined(_STDC_C99) || \
        (!defined(_STRICT_STDC) && !defined(__XOPEN_OR_POSIX)) || \
        defined(__C99FEATURES__)
#if defined(__FLT_EVAL_METHOD__) && __FLT_EVAL_METHOD__ - 0 == 0
typedef float float_t;
typedef double double_t;
#elif __FLT_EVAL_METHOD__ - 0 == 1
typedef double float_t;
typedef double double_t;
#elif __FLT_EVAL_METHOD__ - 0 == 2
typedef long double float_t;
typedef long double double_t;
#elif defined(__sparc) || defined(__amd64)
typedef float float_t;
typedef double double_t;
#elif defined(__i386)
typedef long double float_t;
typedef long double double_t;
#endif

#undef  FP_ZERO
#define FP_ZERO         0
#undef  FP_SUBNORMAL
#define FP_SUBNORMAL    1
#undef  FP_NORMAL
#define FP_NORMAL       2
#undef  FP_INFINITE
#define FP_INFINITE     3
#undef  FP_NAN
#define FP_NAN          4

#undef  FP_ILOGB0
#define FP_ILOGB0       (-2147483647)
#undef  FP_ILOGBNAN
#define FP_ILOGBNAN     2147483647

#undef  MATH_ERRNO
#define MATH_ERRNO      1
#undef  MATH_ERREXCEPT
#define MATH_ERREXCEPT  2
#undef  math_errhandling
#define math_errhandling        MATH_ERREXCEPT

extern double acosh(double);
extern double asinh(double);
extern double atanh(double);

extern double exp2(double);
extern double expm1(double);
extern int ilogb(double);
extern double log1p(double);
extern double log2(double);
extern double logb(double);
extern double scalbn(double, int);
extern double scalbln(double, long int);

extern double cbrt(double);
extern double hypot(double, double);

extern double erf(double);
extern double erfc(double);
extern double lgamma(double);
extern double tgamma(double);

extern double nearbyint(double);
extern double rint(double);
extern long int lrint(double);
extern double round(double);
extern long int lround(double);
extern double trunc(double);

extern double remainder(double, double);
extern double remquo(double, double, int *);

extern double copysign(double, double);
extern double nan(const char *);
extern double nextafter(double, double);
extern double nexttoward(double, long double);

extern double fdim(double, double);
extern double fmax(double, double);
extern double fmin(double, double);

extern double fma(double, double, double);

extern float acosf(float);
extern float asinf(float);
extern float atanf(float);
extern float atan2f(float, float);
extern float cosf(float);
extern float sinf(float);
extern float tanf(float);

extern float acoshf(float);
extern float asinhf(float);
extern float atanhf(float);
extern float coshf(float);
extern float sinhf(float);
extern float tanhf(float);

extern float expf(float);
extern float exp2f(float);
extern float expm1f(float);
extern float frexpf(float, int *);
extern int ilogbf(float);
extern float ldexpf(float, int);
extern float logf(float);
extern float log10f(float);
extern float log1pf(float);
extern float log2f(float);
extern float logbf(float);
extern float modff(float, float *);
extern float scalbnf(float, int);
extern float scalblnf(float, long int);

extern float cbrtf(float);
extern float fabsf(float);
extern float hypotf(float, float);
extern float powf(float, float);
extern float sqrtf(float);

extern float erff(float);
extern float erfcf(float);
extern float lgammaf(float);
extern float tgammaf(float);

extern float ceilf(float);
extern float floorf(float);
extern float nearbyintf(float);
extern float rintf(float);
extern long int lrintf(float);
```

```
extern float roundf(float);
extern long int lroundf(float);
extern float truncf(float);

extern float fmodf(float, float);
extern float remainderf(float, float);
extern float remquof(float, float, int *);

extern float copysignf(float, float);
extern float nanf(const char *);
extern float nextafterf(float, float);
extern float nexttowardf(float, long double);

extern float fdimf(float, float);
extern float fmaxf(float, float);
extern float fminf(float, float);

extern float fmaf(float, float, float);

extern long double acosl(long double);
extern long double asinl(long double);
extern long double atanl(long double);
extern long double atan2l(long double, long double);
extern long double cosl(long double);
extern long double sinl(long double);
extern long double tanl(long double);

extern long double acoshl(long double);
extern long double asinhl(long double);
extern long double atanhl(long double);
extern long double coshl(long double);
extern long double sinhl(long double);
extern long double tanhl(long double);

extern long double expl(long double);
extern long double exp2l(long double);
extern long double expm1l(long double);
extern long double frexpl(long double, int *);
extern int ilogbl(long double);
extern long double ldexpl(long double, int);
extern long double logl(long double);
extern long double log10l(long double);
extern long double log1pl(long double);
extern long double log2l(long double);
extern long double logbl(long double);
extern long double modfl(long double, long double *);
extern long double scalbnl(long double, int);
extern long double scalblnl(long double, long int);

extern long double cbrtl(long double);
extern long double fabsl(long double);
extern long double hypotl(long double, long double);
extern long double powl(long double, long double);
extern long double sqrtl(long double);

extern long double erfl(long double);
extern long double erfcl(long double);
extern long double lgammal(long double);
extern long double tgammal(long double);

extern long double ceill(long double);
extern long double floorl(long double);
extern long double nearbyintl(long double);
extern long double rintl(long double);
extern long int lrintl(long double);
extern long double roundl(long double);
extern long int lroundl(long double);
extern long double truncl(long double);

extern long double fmodl(long double, long double);
extern long double remainderl(long double, long double);
extern long double remquol(long double, long double, int *);

extern long double copysignl(long double, long double);
extern long double nanl(const char *);
extern long double nextafterl(long double, long double);
extern long double nexttowardl(long double, long double);

extern long double fdiml(long double, long double);
extern long double fmaxl(long double, long double);
extern long double fminl(long double, long double);
extern double acosh __P((double));
extern double asinh __P((double));
extern double atanh __P((double));

extern double exp2 __P((double));
extern double expm1 __P((double));
extern int ilogb __P((double));
extern double log1p __P((double));
extern double log2 __P((double));
extern double logb __P((double));
extern double scalbn __P((double, int));
extern double scalbln __P((double, long int));

extern double cbrt __P((double));
extern double hypot __P((double, double));

extern double erf __P((double));
extern double erfc __P((double));
extern double lgamma __P((double));
extern double tgamma __P((double));

extern double nearbyint __P((double));
extern double rint __P((double));
extern long int lrint __P((double));
extern double round __P((double));
extern long int lround __P((double));
extern double trunc __P((double));

extern double remainder __P((double, double));
extern double remquo __P((double, double, int *));

extern double copysign __P((double, double));
extern double nan __P((const char *));
extern double nextafter __P((double, double));
extern double nexttoward __P((double, long double));

extern double fdim __P((double, double));
extern double fmax __P((double, double));
extern double fmin __P((double, double));

extern double fma __P((double, double, double));

extern float acosf __P((float));
extern float asinf __P((float));
extern float atanf __P((float));
extern float atan2f __P((float, float));
extern float cosf __P((float));
extern float sinf __P((float));
extern float tanf __P((float));

extern float acoshf __P((float));
```

```
extern float asinhf __P((float));
extern float atanhf __P((float));
extern float coshf __P((float));
extern float sinhf __P((float));
extern float tanhf __P((float));

extern float expf __P((float));
extern float exp2f __P((float));
extern float expm1f __P((float));
extern float frexpf __P((float, int *));
extern int ilogbf __P((float));
extern float ldexpf __P((float, int));
extern float logf __P((float));
extern float log10f __P((float));
extern float log1pf __P((float));
extern float log2f __P((float));
extern float logbf __P((float));
extern float modff __P((float, float *));
extern float scalbnf __P((float, int));
extern float scalblnf __P((float, long int));

extern float cbrtf __P((float));
extern float fabsf __P((float));
extern float hypotf __P((float, float));
extern float powf __P((float, float));
extern float sqrtf __P((float));

extern float erff __P((float));
extern float erfcf __P((float));
extern float lgammaf __P((float));
extern float tgammaf __P((float));

extern float ceilf __P((float));
extern float floorf __P((float));
extern float nearbyintf __P((float));
extern float rintf __P((float));
extern long int lrintf __P((float));
extern float roundf __P((float));
extern long int lroundf __P((float));
extern float truncf __P((float));

extern float fmodf __P((float, float));
extern float remainderf __P((float, float));
extern float remquof __P((float, float, int *));

extern float copysignf __P((float, float));
extern float nanf __P((const char *));
extern float nextafterf __P((float, float));
extern float nexttowardf __P((float, long double));

extern float fdimf __P((float, float));
extern float fmaxf __P((float, float));
extern float fminf __P((float, float));

extern float fmaf __P((float, float, float));

extern long double acosl __P((long double));
extern long double asinl __P((long double));
extern long double atanl __P((long double));
extern long double atan2l __P((long double, long double));
extern long double cosl __P((long double));
extern long double sinl __P((long double));
extern long double tanl __P((long double));

extern long double acoshl __P((long double));
extern long double asinhl __P((long double));
extern long double atanhl __P((long double));
extern long double coshl __P((long double));
extern long double sinhl __P((long double));
extern long double tanhl __P((long double));

extern long double expl __P((long double));
extern long double exp2l __P((long double));
extern long double expm1l __P((long double));
extern long double frexpl __P((long double, int *));
extern int ilogbl __P((long double));
extern long double ldexpl __P((long double, int));
extern long double logl __P((long double));
extern long double log10l __P((long double));
extern long double log1pl __P((long double));
extern long double log2l __P((long double));
extern long double logbl __P((long double));
extern long double modfl __P((long double, long double *));
extern long double scalbnl __P((long double, int));
extern long double scalblnl __P((long double, long int));

extern long double cbrtl __P((long double));
extern long double fabsl __P((long double));
extern long double hypotl __P((long double, long double));
extern long double powl __P((long double, long double));
extern long double sqrtl __P((long double));

extern long double erfl __P((long double));
extern long double erfcl __P((long double));
extern long double lgammal __P((long double));
extern long double tgammal __P((long double));

extern long double ceill __P((long double));
extern long double floorl __P((long double));
extern long double nearbyintl __P((long double));
extern long double rintl __P((long double));
extern long int lrintl __P((long double));
extern long double roundl __P((long double));
extern long int lroundl __P((long double));
extern long double truncl __P((long double));

extern long double fmodl __P((long double, long double));
extern long double remainderl __P((long double, long double));
extern long double remquol __P((long double, long double, int *));

extern long double copysignl __P((long double, long double));
extern long double nanl __P((const char *));
extern long double nextafterl __P((long double, long double));
extern long double nexttowardl __P((long double, long double));

extern long double fdiml __P((long double, long double));
extern long double fmaxl __P((long double, long double));
extern long double fminl __P((long double, long double));

extern long double fmal(long double, long double, long double);
extern long double fmal __P((long double, long double, long double));

#if !defined(_STRICT_STDC) && !defined(_NO_LONGLONG) || defined(_STDC_C99) || \
        defined(__C99FEATURES__)
extern long long int llrint(double);
extern long long int llround(double);
extern long long int llrint __P((double));
extern long long int llround __P((double));

extern long long int llrintf(float);
extern long long int llroundf(float);
extern long long int llrintf __P((float));
```

```
extern long long int llroundf __P((float));

extern long long int llrintl(long double);
extern long long int llroundl(long double);
extern long long int llrintl __P((long double));
extern long long int llroundl __P((long double));
#endif

#if !defined(__cplusplus)
#pragma does_not_read_global_data(asinh, exp2, expm1)
#pragma does_not_read_global_data(ilogb, log2)
#pragma does_not_read_global_data(scalbn, scalbln, cbrt)
#pragma does_not_read_global_data(erf, erfc, tgamma)
#pragma does_not_read_global_data(nearbyint, rint, lrint, round, lround, trunc)
#pragma does_not_read_global_data(remquo)
#pragma does_not_read_global_data(copysign, nan, nexttoward)
#pragma does_not_read_global_data(fdim, fmax, fmin, fma)
#pragma does_not_write_global_data(asinh, exp2, expm1)
#pragma does_not_write_global_data(ilogb, log2)
#pragma does_not_write_global_data(scalbn, scalbln, cbrt)
#pragma does_not_write_global_data(erf, erfc, tgamma)
#pragma does_not_write_global_data(nearbyint, rint, lrint, round, lround, trunc)
#pragma does_not_write_global_data(copysign, nan, nexttoward)
#pragma does_not_write_global_data(fdim, fmax, fmin, fma)

#pragma does_not_read_global_data(acosf, asinf, atanf, atan2f)
#pragma does_not_read_global_data(cosf, sinf, tanf)
#pragma does_not_read_global_data(acoshf, asinhf, atanhf, coshf, sinhf, tanhf)
#pragma does_not_read_global_data(expf, exp2f, expm1f, frexpf, ilogbf, ldexpf)
#pragma does_not_read_global_data(logf, log10f, log1pf, log2f, logbf)
#pragma does_not_read_global_data(modff, scalbnf, scalblnf)
#pragma does_not_read_global_data(cbrtf, fabsf, hypotf, powf, sqrtf)
#pragma does_not_read_global_data(erff, erfcf, lgammaf, tgammaf)
#pragma does_not_read_global_data(ceilf, floorf, nearbyintf)
#pragma does_not_read_global_data(rintf, lrintf, roundf, lroundf, truncf)
#pragma does_not_read_global_data(fmodf, remainderf, remquof)
#pragma does_not_read_global_data(copysignf, nanf, nextafterf, nexttowardf)
#pragma does_not_read_global_data(fdimf, fmaxf, fminf, fmaf)
#pragma does_not_write_global_data(acosf, asinf, atanf, atan2f)
#pragma does_not_write_global_data(cosf, sinf, tanf)
#pragma does_not_write_global_data(acoshf, asinhf, atanhf, coshf, sinhf, tanhf)
#pragma does_not_write_global_data(expf, exp2f, expm1f, ilogbf, ldexpf)
#pragma does_not_write_global_data(logf, log10f, log1pf, log2f, logbf)
#pragma does_not_write_global_data(cbrtf, fabsf, hypotf, powf, sqrtf)
#pragma does_not_write_global_data(erff, erfcf, tgammaf)
#pragma does_not_write_global_data(ceilf, floorf, nearbyintf)
#pragma does_not_write_global_data(rintf, lrintf, roundf, lroundf, truncf)
#pragma does_not_write_global_data(fmodf, remainderf)
#pragma does_not_write_global_data(copysignf, nanf, nextafterf, nexttowardf)
#pragma does_not_write_global_data(fdimf, fmaxf, fminf, fmaf)

#pragma does_not_read_global_data(acosl, asinl, atanl, atan2l)
#pragma does_not_read_global_data(cosl, sinl, tanl)
#pragma does_not_read_global_data(acoshl, asinhl, atanhl, coshl, sinhl, tanhl)
#pragma does_not_read_global_data(expl, exp2l, expm1l, frexpl, ilogbl, ldexpl)
#pragma does_not_read_global_data(logl, log10l, log1pl, log2l, logbl)
#pragma does_not_read_global_data(modfl, scalbnl, scalblnl)
#pragma does_not_read_global_data(cbrtl, fabsl, hypotl, powl, sqrtl)
#pragma does_not_read_global_data(erfl, erfcl, lgammal, tgammal)
#pragma does_not_read_global_data(ceill, floorl, nearbyintl)
#pragma does_not_read_global_data(rintl, lrintl, roundl, lroundl, truncl)
#pragma does_not_read_global_data(fmodl, remainderl, remquol)
#pragma does_not_read_global_data(copysignl, nanl, nextafterl, nexttowardl)
#pragma does_not_read_global_data(fdiml, fmaxl, fminl, fmal)
#pragma does_not_write_global_data(acosl, asinl, atanl, atan2l)
#pragma does_not_write_global_data(cosl, sinl, tanl)
```

```
#pragma does_not_write_global_data(acoshl, asinhl, atanhl, coshl, sinhl, tanhl)
#pragma does_not_write_global_data(expl, exp2l, expm1l, ilogbl, ldexpl)
#pragma does_not_write_global_data(logl, log10l, log1pl, log2l, logbl)
#pragma does_not_write_global_data(cbrtl, fabsl, hypotl, powl, sqrtl)
#pragma does_not_write_global_data(erfl, erfcl, tgammal)
#pragma does_not_write_global_data(ceill, floorl, nearbyintl)
#pragma does_not_write_global_data(rintl, lrintl, roundl, lroundl, truncl)
#pragma does_not_write_global_data(fmodl, remainderl)
#pragma does_not_write_global_data(copysignl, nanl, nextafterl, nexttowardl)
#pragma does_not_write_global_data(fdiml, fmaxl, fminl, fmal)

#if !defined(_STRICT_STDC) && !defined(_NO_LONGLONG) || defined(_STDC_C99) || \
        defined(__C99FEATURES__)
#pragma does_not_read_global_data(llrint, llround)
#pragma does_not_read_global_data(llrintf, llroundf, llrintl, llroundl)
#pragma does_not_write_global_data(llrint, llround)
#pragma does_not_write_global_data(llrintf, llroundf, llrintl, llroundl)
#endif
#endif  /* !defined(__cplusplus) */

#if defined(__MATHERR_ERRNO_DONTCARE)
#pragma does_not_read_global_data(acosh, atanh, hypot, lgamma, log1p, logb)
#pragma does_not_read_global_data(nextafter, remainder)
#pragma does_not_write_global_data(acosh, atanh, hypot, log1p, logb)
#pragma does_not_write_global_data(nextafter, remainder)

#pragma no_side_effect(acosh, asinh, atanh, exp2, expm1)
#pragma no_side_effect(ilogb, log1p, log2, logb)
#pragma no_side_effect(scalbn, scalbln, cbrt, hypot)
#pragma no_side_effect(erf, erfc, tgamma)
#pragma no_side_effect(nearbyint, rint, lrint, round, lround, trunc)
#pragma no_side_effect(remainder)
#pragma no_side_effect(copysign, nan, nextafter, nexttoward)
#pragma no_side_effect(fdim, fmax, fmin, fma)

#pragma no_side_effect(acosf, asinf, atanf, atan2f)
#pragma no_side_effect(cosf, sinf, tanf, coshf, sinhf, tanhf)
#pragma no_side_effect(acoshf, asinhf, atanhf, coshf, sinhf, tanhf)
#pragma no_side_effect(expf, exp2f, expm1f, ilogbf, ldexpf)
#pragma no_side_effect(logf, log10f, log1pf, log2f, logbf)
#pragma no_side_effect(cbrtf, fabsf, hypotf, powf, sqrtf)
#pragma no_side_effect(erff, erfcf, tgammaf)
#pragma no_side_effect(ceilf, floorf, nearbyintf)
#pragma no_side_effect(rintf, lrintf, roundf, lroundf, truncf)
#pragma no_side_effect(fmodf, remainderf)
#pragma no_side_effect(copysignf, nanf, nextafterf, nexttowardf)
#pragma no_side_effect(fdimf, fmaxf, fminf, fmaf)

#pragma no_side_effect(acosl, asinl, atanl, atan2l)
#pragma no_side_effect(cosl, sinl, tanl, coshl, sinhl, tanhl)
#pragma no_side_effect(acoshl, asinhl, atanhl, coshl, sinhl, tanhl)
#pragma no_side_effect(expl, exp2l, expm1l, ilogbl, ldexpl)
#pragma no_side_effect(logl, log10l, log1pl, log2l, logbl)
#pragma no_side_effect(cbrtl, fabsl, hypotl, powl, sqrtl)
#pragma no_side_effect(erfl, erfcl, tgammal)
#pragma no_side_effect(ceill, floorl, nearbyintl)
#pragma no_side_effect(rintl, lrintl, roundl, lroundl, truncl)
#pragma no_side_effect(fmodl, remainderl)
#pragma no_side_effect(copysignl, nanl, nextafterl, nexttowardl)
#pragma no_side_effect(fdiml, fmaxl, fminl, fmal)

#if !defined(_STRICT_STDC) && !defined(_NO_LONGLONG) || defined(_STDC_C99) || \
        defined(__C99FEATURES__)
#pragma no_side_effect(llrint, llround, llrintf, llroundf, llrintl, llroundl)
#endif
#endif  /* defined(__MATHERR_ERRNO_DONTCARE) */
```

```
#endif  /* defined(__EXTENSIONS__) || defined(_STDC_C99) || ... */

#ifdef __cplusplus
}
```

______***unchanged_portion_omitted_***

```
**********************************************************
    8375 Tue Apr 30 23:28:35 2019
new/usr/src/head/iso/math_iso.h
10882 math headers should stop supporting K&R C
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
 */
/*
 * Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _ISO_MATH_ISO_H
#define _ISO_MATH_ISO_H

#include <sys/feature_tests.h>

#ifdef __cplusplus
extern "C" {
#endif

#ifndef __P
#ifdef __STDC__
#define __P(p)  p
#else
#define __P(p)  ()
#endif
#endif  /* !defined(__P) */

#if !defined(_STDC_C99) && _XOPEN_SOURCE - 0 < 600 && !defined(__C99FEATURES__)
typedef union _h_val {
        unsigned long _i[sizeof (double) / sizeof (unsigned long)];
        double _d;
} _h_val;

#ifdef __STDC__
extern const _h_val __huge_val;
#else
extern _h_val __huge_val;
#endif
#undef  HUGE_VAL
#define HUGE_VAL __huge_val._d
#endif  /* !defined(_STDC_C99) && _XOPEN_SOURCE - 0 < 600 && ... */

#if __cplusplus >= 199711L
```

```
namespace std {
#endif

extern double acos(double);
extern double asin(double);
extern double atan(double);
extern double atan2(double, double);
extern double cos(double);
extern double sin(double);
extern double tan(double);

extern double cosh(double);
extern double sinh(double);
extern double tanh(double);

extern double exp(double);
extern double frexp(double, int *);
extern double ldexp(double, int);
extern double log(double);
extern double log10(double);
extern double modf(double, double *);

extern double pow(double, double);
extern double sqrt(double);

extern double ceil(double);
extern double fabs(double);
extern double floor(double);
extern double fmod(double, double);
extern double acos __P((double));
extern double asin __P((double));
extern double atan __P((double));
extern double atan2 __P((double, double));
extern double cos __P((double));
extern double sin __P((double));
extern double tan __P((double));

extern double cosh __P((double));
extern double sinh __P((double));
extern double tanh __P((double));

extern double exp __P((double));
extern double frexp __P((double, int *));
extern double ldexp __P((double, int));
extern double log __P((double));
extern double log10 __P((double));
extern double modf __P((double, double *));

extern double pow __P((double, double));
extern double sqrt __P((double));

extern double ceil __P((double));
extern double fabs __P((double));
extern double floor __P((double));
extern double fmod __P((double, double));

#if defined(__MATHERR_ERRNO_DONTCARE)
#pragma does_not_read_global_data(acos, asin, atan, atan2)
#pragma does_not_read_global_data(cos, sin, tan, cosh, sinh, tanh)
#pragma does_not_read_global_data(exp, log, log10, pow, sqrt)
#pragma does_not_read_global_data(frexp, ldexp, modf)
#pragma does_not_read_global_data(ceil, fabs, floor, fmod)
#pragma does_not_write_global_data(acos, asin, atan, atan2)
#pragma does_not_write_global_data(cos, sin, tan, cosh, sinh, tanh)
#pragma does_not_write_global_data(exp, log, log10, pow, sqrt)
#pragma does_not_write_global_data(ldexp)
```

```
#pragma does_not_write_global_data(ceil, fabs, floor, fmod)
#pragma no_side_effect(acos, asin, atan, atan2)
#pragma no_side_effect(cos, sin, tan, cosh, sinh, tanh)
#pragma no_side_effect(exp, log, log10, pow, sqrt)
#pragma no_side_effect(ldexp)
#pragma no_side_effect(ceil, fabs, floor, fmod)
#endif

#if __cplusplus >= 199711L
extern float __acosf(float);
extern float __asinf(float);
extern float __atanf(float);
extern float __atan2f(float, float);
extern float __ceilf(float);
extern float __cosf(float);
extern float __coshf(float);
extern float __expf(float);
extern float __fabsf(float);
extern float __floorf(float);
extern float __fmodf(float, float);
extern float __frexpf(float, int *);
extern float __ldexpf(float, int);
extern float __logf(float);
extern float __log10f(float);
extern float __modff(float, float *);
extern float __powf(float, float);
extern float __sinf(float);
extern float __sinhf(float);
extern float __sqrtf(float);
extern float __tanf(float);
extern float __tanhf(float);

extern long double __acosl(long double);
extern long double __asinl(long double);
extern long double __atanl(long double);
extern long double __atan2l(long double, long double);
extern long double __ceill(long double);
extern long double __cosl(long double);
extern long double __coshl(long double);
extern long double __expl(long double);
extern long double __fabsl(long double);
extern long double __floorl(long double);
extern long double __fmodl(long double, long double);
extern long double __frexpl(long double, int *);
extern long double __ldexpl(long double, int);
extern long double __logl(long double);
extern long double __log10l(long double);
extern long double __modfl(long double, long double *);
extern long double __powl(long double, long double);
extern long double __sinl(long double);
extern long double __sinhl(long double);
extern long double __sqrtl(long double);
extern long double __tanl(long double);
extern long double __tanhl(long double);

extern "C++" {
#undef  __X
#undef  __Y
        inline double abs(double __X) { return fabs(__X); }

        inline double pow(double __X, int __Y) {
                return (pow(__X, (double)(__Y)));
        }
        inline double pow(double __X, int __Y) { return
                pow(__X, (double) (__Y)); }
```

```
        inline float abs(float __X) { return __fabsf(__X); }
        inline float acos(float __X) { return __acosf(__X); }
        inline float asin(float __X) { return __asinf(__X); }
        inline float atan(float __X) { return __atanf(__X); }
        inline float atan2(float __X, float __Y) { return __atan2f(__X, __Y); }
        inline float ceil(float __X) { return __ceilf(__X); }
        inline float cos(float __X) { return __cosf(__X); }
        inline float cosh(float __X) { return __coshf(__X); }
        inline float exp(float __X) { return __expf(__X); }
        inline float fabs(float __X) { return __fabsf(__X); }
        inline float floor(float __X) { return __floorf(__X); }
        inline float fmod(float __X, float __Y) { return __fmodf(__X, __Y); }
        inline float frexp(float __X, int *__Y) { return __frexpf(__X, __Y); }
        inline float ldexp(float __X, int __Y) { return __ldexpf(__X, __Y); }
        inline float log(float __X) { return __logf(__X); }
        inline float log10(float __X) { return __log10f(__X); }
        inline float modf(float __X, float *__Y) { return __modff(__X, __Y); }
        inline float pow(float __X, float __Y) { return __powf(__X, __Y); }

        inline float pow(float __X, int __Y) {
                return (pow((double)(__X), (double)(__Y)));
        }

        inline float pow(float __X, int __Y) { return
                pow((double) (__X), (double) (__Y)); }
        inline float sin(float __X) { return __sinf(__X); }
        inline float sinh(float __X) { return __sinhf(__X); }
        inline float sqrt(float __X) { return __sqrtf(__X); }
        inline float tan(float __X) { return __tanf(__X); }
        inline float tanh(float __X) { return __tanhf(__X); }

        inline long double abs(long double __X) { return __fabsl(__X); }
        inline long double acos(long double __X) { return __acosl(__X); }
        inline long double asin(long double __X) { return __asinl(__X); }
        inline long double atan(long double __X) { return __atanl(__X); }

        inline long double atan2(long double __X, long double __Y) {
                return (__atan2l(__X, __Y));
        }

        inline long double atan2(long double __X, long double __Y) { return
                __atan2l(__X, __Y); }
        inline long double ceil(long double __X) { return __ceill(__X); }
        inline long double cos(long double __X) { return __cosl(__X); }
        inline long double cosh(long double __X) { return __coshl(__X); }
        inline long double exp(long double __X) { return __expl(__X); }
        inline long double fabs(long double __X) { return __fabsl(__X); }
        inline long double floor(long double __X) { return __floorl(__X); }

        inline long double fmod(long double __X, long double __Y) {
                return (__fmodl(__X, __Y));
        }

        inline long double frexp(long double __X, int *__Y) {
                return (__frexpl(__X, __Y));
        }

        inline long double ldexp(long double __X, int __Y) {
                return (__ldexpl(__X, __Y));
        }

        inline long double fmod(long double __X, long double __Y) { return
                __fmodl(__X, __Y); }
        inline long double frexp(long double __X, int *__Y) { return
                __frexpl(__X, __Y); }
        inline long double ldexp(long double __X, int __Y) { return
```

```
                __ldexpl(__X, __Y); }
        inline long double log(long double __X) { return __logl(__X); }
        inline long double log10(long double __X) { return __log10l(__X); }

        inline long double modf(long double __X, long double *__Y) {
                return (__modfl(__X, __Y));
        }

        inline long double pow(long double __X, long double __Y) {
                return (__powl(__X, __Y));
        }

        inline long double pow(long double __X, int __Y) {
                return (__powl(__X, (long double) (__Y)));
        }

        inline long double modf(long double __X, long double *__Y) { return
                __modfl(__X, __Y); }
        inline long double pow(long double __X, long double __Y) { return
                __powl(__X, __Y); }
        inline long double pow(long double __X, int __Y) { return
                __powl(__X, (long double) (__Y)); }
        inline long double sin(long double __X) { return __sinl(__X); }
        inline long double sinh(long double __X) { return __sinhl(__X); }
        inline long double sqrt(long double __X) { return __sqrtl(__X); }
        inline long double tan(long double __X) { return __tanl(__X); }
        inline long double tanh(long double __X) { return __tanhl(__X); }
}       /* end of extern "C++" */
```

______***unchanged_portion_omitted_***

```
**********************************************************
    9976 Tue Apr 30 23:28:36 2019
new/usr/src/head/math.h
10882 math headers should stop supporting K&R C
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2011 Nexenta Systems, Inc.  All rights reserved.
 */
/*
 * Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _MATH_H
#define _MATH_H

#include <iso/math_iso.h>
#include <iso/math_c99.h>

#if __cplusplus >= 199711L
using std::abs;
using std::acos;
using std::asin;
using std::atan2;
using std::atan;
using std::ceil;
using std::cos;
using std::cosh;
using std::exp;
using std::fabs;
using std::floor;
using std::fmod;
using std::frexp;
using std::ldexp;
using std::log10;
using std::log;
using std::modf;
using std::pow;
using std::sin;
using std::sinh;
using std::sqrt;
using std::tan;
using std::tanh;
#endif

#ifdef __cplusplus
```

```
extern "C" {
#endif

#if defined(__cplusplus)
#define exception       __math_exception
#endif

#ifndef __P
#ifdef __STDC__
#define __P(p)  p
#else
#define __P(p)  ()
#endif
#endif  /* !defined(__P) */

#if defined(__EXTENSIONS__) || defined(_XOPEN_SOURCE) || \
        !defined(_STRICT_STDC) && !defined(_POSIX_C_SOURCE)
/*
 * SVID & X/Open
 */
#define M_E             2.7182818284590452354
#define M_LOG2E         1.4426950408889634074
#define M_LOG10E        0.43429448190325182765
#define M_LN2           0.69314718055994530942
#define M_LN10          2.30258509299404568402
#define M_PI            3.14159265358979323846
#define M_PI_2          1.57079632679489661923
#define M_PI_4          0.78539816339744830962
#define M_1_PI          0.31830988618379067154
#define M_2_PI          0.63661977236758134308
#define M_2_SQRTPI      1.12837916709551257390
#define M_SQRT2         1.41421356237309504880
#define M_SQRT1_2       0.70710678118654752440

extern int signgam;

#define MAXFLOAT        ((float)3.40282346638528860e+38)

#if defined(__EXTENSIONS__) || !defined(_XOPEN_SOURCE)
/*
 * SVID
 */
enum version {libm_ieee = -1, c_issue_4, ansi_1, strict_ansi};

#ifdef __STDC__
extern const enum version _lib_version;
#else
extern enum version _lib_version;
#endif

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

#define HUGE            MAXFLOAT

#define _ABS(x)         ((x) < 0 ? -(x) : (x))

#define _REDUCE(TYPE, X, XN, C1, C2)    { \
        double x1 = (double)(TYPE)X, x2 = X - x1; \
        X = x1 - (XN) * (C1); X += x2; X -= (XN) * (C2); }
```

```
#define DOMAIN          1
#define SING            2
#define OVERFLOW        3
#define UNDERFLOW       4
#define TLOSS           5
#define PLOSS           6

#define _POLY1(x, c)    ((c)[0] * (x) + (c)[1])
#define _POLY2(x, c)    (_POLY1((x), (c)) * (x) + (c)[2])
#define _POLY3(x, c)    (_POLY2((x), (c)) * (x) + (c)[3])
#define _POLY4(x, c)    (_POLY3((x), (c)) * (x) + (c)[4])
#define _POLY5(x, c)    (_POLY4((x), (c)) * (x) + (c)[5])
#define _POLY6(x, c)    (_POLY5((x), (c)) * (x) + (c)[6])
#define _POLY7(x, c)    (_POLY6((x), (c)) * (x) + (c)[7])
#define _POLY8(x, c)    (_POLY7((x), (c)) * (x) + (c)[8])
#define _POLY9(x, c)    (_POLY8((x), (c)) * (x) + (c)[9])
#endif  /* defined(__EXTENSIONS__) || !defined(_XOPEN_SOURCE) */

/*
 * SVID & X/Open
 */
/* BEGIN adopted by C99 */
extern double erf(double);
extern double erfc(double);
extern double hypot(double, double);
extern double lgamma(double);
extern double erf __P((double));
extern double erfc __P((double));
extern double hypot __P((double, double));
extern double lgamma __P((double));

#if defined(__MATHERR_ERRNO_DONTCARE)
#pragma does_not_read_global_data(erf, erfc, hypot)
#pragma does_not_write_global_data(erf, erfc, hypot)
#pragma no_side_effect(erf, erfc, hypot)
#endif

#if !defined(_STDC_C99) && _XOPEN_SOURCE - 0 < 600 && !defined(__C99FEATURES__)
extern int isnan(double);
extern int isnan __P((double));

#pragma does_not_read_global_data(isnan)
#pragma does_not_write_global_data(isnan)
#pragma no_side_effect(isnan)
#endif
/* END adopted by C99 */

#if defined(__EXTENSIONS__) || _XOPEN_SOURCE - 0 < 600
extern double gamma(double);            /* deprecated; use lgamma */
extern double gamma __P((double));              /* deprecated; use lgamma */
#endif
extern double j0(double);
extern double j1(double);
extern double jn(int, double);
extern double y0(double);
extern double y1(double);
extern double yn(int, double);
extern double j0 __P((double));
extern double j1 __P((double));
extern double jn __P((int, double));
extern double y0 __P((double));
extern double y1 __P((double));
extern double yn __P((int, double));

#if defined(__MATHERR_ERRNO_DONTCARE)
#pragma does_not_read_global_data(j0, j1, jn, y0, y1, yn)
```

```
#pragma does_not_write_global_data(j0, j1, jn, y0, y1, yn)
#pragma no_side_effect(j0, j1, jn, y0, y1, yn)
#endif
#if defined(__EXTENSIONS__) || !defined(_XOPEN_SOURCE) || \
        _XOPEN_SOURCE - 0 >= 500 || \
        defined(_XOPEN_SOURCE) && _XOPEN_SOURCE_EXTENDED - 0 == 1
/*
 * SVID & XPG 4.2/5
 */
extern double scalb(double, double);
extern double scalb __P((double, double));

#if defined(__MATHERR_ERRNO_DONTCARE)
#pragma does_not_read_global_data(scalb)
#pragma does_not_write_global_data(scalb)
#pragma no_side_effect(scalb)
#endif

/* BEGIN adopted by C99 */
extern double acosh(double);
extern double asinh(double);
extern double atanh(double);
extern double cbrt(double);
extern double logb(double);
extern double nextafter(double, double);
extern double remainder(double, double);
extern double acosh __P((double));
extern double asinh __P((double));
extern double atanh __P((double));
extern double cbrt __P((double));
extern double logb __P((double));
extern double nextafter __P((double, double));
extern double remainder __P((double, double));

/*
 * XPG 4.2/5
 */
extern double expm1(double);
extern int ilogb(double);
extern double log1p(double);
extern double rint(double);
extern double expm1 __P((double));
extern int ilogb __P((double));
extern double log1p __P((double));
extern double rint __P((double));

#if defined(__MATHERR_ERRNO_DONTCARE)
#pragma does_not_read_global_data(acosh, asinh, atanh, cbrt)
#pragma does_not_read_global_data(logb, nextafter, remainder)
#pragma does_not_read_global_data(expm1, ilogb, log1p, rint)
#pragma does_not_write_global_data(acosh, asinh, atanh, cbrt)
#pragma does_not_write_global_data(logb, nextafter, remainder)
#pragma does_not_write_global_data(expm1, ilogb, log1p, rint)
#pragma no_side_effect(acosh, asinh, atanh, cbrt)
#pragma no_side_effect(logb, nextafter, remainder)
#pragma no_side_effect(expm1, ilogb, log1p, rint)
#endif
/* END adopted by C99 */
#endif  /* defined(__EXTENSIONS__) || !defined(_XOPEN_SOURCE) || ... */

#if defined(__EXTENSIONS__) || !defined(_XOPEN_SOURCE)
/*
 * SVID
 */
extern int matherr(struct exception *);
extern int matherr __P((struct exception *));
```

```
/*
 * IEEE Test Vector
 */
extern double significand(double);
extern double significand __P((double));

#if defined(__MATHERR_ERRNO_DONTCARE)
#pragma does_not_read_global_data(significand)
#pragma does_not_write_global_data(significand)
#pragma no_side_effect(significand)
#endif

extern int signgamf;                            /* deprecated; use signgam */
extern int signgaml;                            /* deprecated; use signgam */

extern int isnanf(float);
extern int isnanl(long double);
extern float gammaf(float);             /* deprecated; use lgammaf */
extern float gammaf_r(float, int *);    /* deprecated; use lgammaf_r */
extern float j0f(float);
extern float j1f(float);
extern float jnf(int, float);
extern float lgammaf_r(float, int *);
extern float scalbf(float, float);
extern float significandf(float);
extern float y0f(float);
extern float y1f(float);
extern float ynf(int, float);
extern long double gammal(long double); /* deprecated; use lgammal */
extern long double gammal_r(long double, int *);        /* deprecated */
extern long double j0l(long double);
extern long double j1l(long double);
extern long double jnl(int, long double);
extern long double lgammal_r(long double, int *);
extern long double scalbl(long double, long double);
extern long double significandl(long double);
extern long double y0l(long double);
extern long double y1l(long double);
extern long double ynl(int, long double);
extern int isnanf __P((float));
extern int isnanl __P((long double));
extern float gammaf __P((float));               /* deprecated; use lgammaf */
extern float gammaf_r __P((float, int *));      /* deprecated; use lgammaf_r */
extern float j0f __P((float));
extern float j1f __P((float));
extern float jnf __P((int, float));
extern float lgammaf_r __P((float, int *));
extern float scalbf __P((float, float));
extern float significandf __P((float));
extern float y0f __P((float));
extern float y1f __P((float));
extern float ynf __P((int, float));
extern long double gammal __P((long double));   /* deprecated; use lgammal */
extern long double gammal_r __P((long double, int *));  /* deprecated */
extern long double j0l __P((long double));
extern long double j1l __P((long double));
extern long double jnl __P((int, long double));
extern long double lgammal_r __P((long double, int *));
extern long double scalbl __P((long double, long double));
extern long double significandl __P((long double));
extern long double y0l __P((long double));
extern long double y1l __P((long double));
extern long double ynl __P((int, long double));

#if defined(__MATHERR_ERRNO_DONTCARE)
```

```
#pragma does_not_read_global_data(isnanf, isnanl)
#pragma does_not_write_global_data(isnanf, isnanl)
#pragma no_side_effect(isnanf, isnanl)
#pragma does_not_read_global_data(gammaf_r, j0f, j1f, jnf, lgammaf_r, scalbf)
#pragma does_not_read_global_data(significandf, y0f, y1f, ynf)
#pragma does_not_write_global_data(j0f, j1f, jnf, scalbf)
#pragma does_not_write_global_data(significandf, y0f, y1f, ynf)
#pragma no_side_effect(j0f, j1f, jnf, scalbf)
#pragma no_side_effect(significandf, y0f, y1f, ynf)
#pragma does_not_read_global_data(gammal_r, j0l, j1l, jnl, lgammal_r, scalbl)
#pragma does_not_read_global_data(significandl, y0l, y1l, ynl)
#pragma does_not_write_global_data(j0l, j1l, jnl, scalbl)
#pragma does_not_write_global_data(significandl, y0l, y1l, ynl)
#pragma no_side_effect(j0l, j1l, jnl, scalbl)
#pragma no_side_effect(significandl, y0l, y1l, ynl)
#endif

/*
 * for sin+cos->sincos transformation
 */
extern void sincos(double, double *, double *);
extern void sincosf(float, float *, float *);
extern void sincosl(long double, long double *, long double *);
extern void sincos __P((double, double *, double *));
extern void sincosf __P((float, float *, float *));
extern void sincosl __P((long double, long double *, long double *));

#if defined(__MATHERR_ERRNO_DONTCARE)
#pragma does_not_read_global_data(sincos, sincosf, sincosl)
#endif

/* BEGIN adopted by C99 */
/*
 * Functions callable from C, intended to support IEEE arithmetic.
 */
extern double copysign(double, double);
extern double scalbn(double, int);
extern double copysign __P((double, double));
extern double scalbn __P((double, int));

#if defined(__MATHERR_ERRNO_DONTCARE)
#pragma does_not_read_global_data(copysign, scalbn)
#pragma does_not_write_global_data(copysign, scalbn)
#pragma no_side_effect(copysign, scalbn)
#endif
/* END adopted by C99 */

/*
 * Reentrant version of gamma & lgamma; passes signgam back by reference
 * as the second argument; user must allocate space for signgam.
 */
extern double gamma_r(double, int *);   /* deprecated; use lgamma_r */
extern double lgamma_r(double, int *);
extern double gamma_r __P((double, int *));     /* deprecated; use lgamma_r */
extern double lgamma_r __P((double, int *));

#if defined(__MATHERR_ERRNO_DONTCARE)
#pragma does_not_read_global_data(gamma_r, lgamma_r)
#endif

/* BEGIN adopted by C99 */
extern float modff(float, float *);
extern float modff __P((float, float *));

#if defined(__MATHERR_ERRNO_DONTCARE)
#pragma does_not_read_global_data(modff)
```

```
#endif
/* END adopted by C99 */

#if defined(__EXTENSIONS__) || !defined(__cplusplus)
#include <floatingpoint.h>
#endif
#endif  /* defined(__EXTENSIONS__) || !defined(_XOPEN_SOURCE) */
#endif  /* defined(__EXTENSIONS__) || defined(_XOPEN_SOURCE) || ... */

#if defined(__cplusplus) && defined(__GNUC__)
#undef  exception
#endif

#ifdef __cplusplus
}
```
______***unchanged_portion_omitted_***