```
**********************************************************
    6743 Sun Mar  3 01:33:06 2019
new/usr/src/head/floatingpoint.h
10495 libc should know how many FPU exceptions there are
**********************************************************
/*
 * 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

#define N_IEEE_EXCEPTION 5      /* Number of floating-point exceptions. */

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 __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_
```

```
**********************************************************
    2786 Sun Mar  3 01:33:07 2019
new/usr/src/uts/common/sys/ieeefp.h
10495 libc should know how many FPU exceptions there are
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (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 2005 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _SYS_IEEEFP_H
#define _SYS_IEEEFP_H

#pragma ident   "%Z%%M% %I%     %E% SMI"        /* SunOS4.0 1.6 */

#ifdef  __cplusplus
extern "C" {
#endif

/*
 * Sun types for IEEE floating point.
 */

#if defined(__sparc)

enum fp_direction_type {        /* rounding direction */
        fp_nearest      = 0,
        fp_tozero       = 1,
        fp_positive     = 2,
        fp_negative     = 3
};
```

*unchanged_portion_omitted_*

```
#define N_IEEE_EXCEPTION 5      /* Number of floating-point exceptions. */
#endif /* ! codereview */

enum fp_trap_enable_type {      /* trap enable bits according to bit number */
        fp_trap_inexact = 0,
        fp_trap_division        = 1,
        fp_trap_underflow       = 2,
        fp_trap_overflow        = 3,
        fp_trap_invalid = 4
};

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



```
enum fp_direction_type {        /* rounding direction */
        fp_nearest      = 0,
        fp_negative     = 1,
        fp_positive     = 2,
        fp_tozero       = 3
};

enum fp_precision_type {        /* extended rounding precision */
        fp_single       = 0,
        fp_precision_3  = 1,
        fp_double       = 2,
        fp_extended     = 3
};

enum fp_exception_type {        /* exceptions according to bit number */
        fp_invalid      = 0,
        fp_denormalized = 1,
        fp_division     = 2,
        fp_overflow     = 3,
        fp_underflow    = 4,
        fp_inexact      = 5
};
#define N_IEEE_EXCEPTION 6      /* Number of floating-point exceptions. */
#endif /* ! codereview */

enum fp_trap_enable_type {      /* trap enable bits according to bit number */
        fp_trap_invalid = 0,
        fp_trap_denormalized    = 1,
        fp_trap_division        = 2,
        fp_trap_overflow        = 3,
        fp_trap_underflow       = 4,
        fp_trap_inexact = 5
};

#endif  /* __i386 || __amd64 */

enum fp_class_type {            /* floating-point classes */
        fp_zero         = 0,
        fp_subnormal    = 1,
        fp_normal       = 2,
        fp_infinity     = 3,
        fp_quiet        = 4,
        fp_signaling    = 5
};

#ifdef  __cplusplus
}
#endif

#endif  /* _SYS_IEEEFP_H */
```