```
**********************************************************
   14668 Fri Jun 21 17:40:58 2013
new/usr/src/uts/common/sys/feature_tests.h
Be more consistent with other OSes. Try to avoid failing on unknown _XOPEN_SOURC
3801 _XOPEN_SOURCE value is ignored if __XOPEN_SOURCE_EXTENDED is set
**********************************************************
```

```
/*
 * 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 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#ifndef _SYS_FEATURE_TESTS_H
#define _SYS_FEATURE_TESTS_H

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

#ifdef  __cplusplus
extern "C" {
#endif

/*
 * Values of _POSIX_C_SOURCE
 *
 *              undefined   not a POSIX compilation
 *              1           POSIX.1-1990 compilation
 *              2           POSIX.2-1992 compilation
 *              199309L     POSIX.1b-1993 compilation (Real Time)
 *              199506L     POSIX.1c-1995 compilation (POSIX Threads)
 *              200112L     POSIX.1-2001 compilation (Austin Group Revision)
 */
#if defined(_POSIX_SOURCE) && !defined(_POSIX_C_SOURCE)
#define _POSIX_C_SOURCE 1
#endif

/*
 * The feature test macros __XOPEN_OR_POSIX, _STRICT_STDC, and _STDC_C99
 * are Sun implementation specific macros created in order to compress
 * common standards specified feature test macros for easier reading.
 * These macros should not be used by the application developer as
 * unexpected results may occur. Instead, the user should reference
 * standards(5) for correct usage of the standards feature test macros.
 *
 * __XOPEN_OR_POSIX     Used in cases where a symbol is defined by both
 *                      X/Open or POSIX or in the negative, when neither
```



```
 *                      X/Open or POSIX defines a symbol.
 *
 * _STRICT_STDC         __STDC__ is specified by the C Standards and defined
 *                      by the compiler. For Sun compilers the value of
 *                      __STDC__ is either 1, 0, or not defined based on the
 *                      compilation mode (see cc(1)). When the value of
 *                      __STDC__ is 1 and in the absence of any other feature
 *                      test macros, the namespace available to the application
 *                      is limited to only those symbols defined by the C
 *                      Standard. _STRICT_STDC provides a more readable means
 *                      of identifying symbols defined by the standard, or in
 *                      the negative, symbols that are extensions to the C
 *                      Standard. See additional comments for GNU C differences.
 *
 * _STDC_C99            __STDC_VERSION__ is specified by the C standards and
 *                      defined by the compiler and indicates the version of
 *                      the C standard. A value of 199901L indicates a
 *                      compiler that complies with ISO/IEC 9899:1999, other-
 *                      wise known as the C99 standard.
 */

#if defined(_XOPEN_SOURCE) || defined(_POSIX_C_SOURCE)
#define __XOPEN_OR_POSIX
#endif

/*
 * ISO/IEC 9899:1990 and it's revision, ISO/IEC 9899:1999 specify the
 * following predefined macro name:
 *
 * __STDC__     The integer constant 1, intended to indicate a conforming
 *              implementation.
 *
 * Furthermore, a strictly conforming program shall use only those features
 * of the language and library specified in these standards. A conforming
 * implementation shall accept any strictly conforming program.
 *
 * Based on these requirements, Sun's C compiler defines __STDC__ to 1 for
 * strictly conforming environments and __STDC__ to 0 for environments that
 * use ANSI C semantics but allow extensions to the C standard. For non-ANSI
 * C semantics, Sun's C compiler does not define __STDC__.
 *
 * The GNU C project interpretation is that __STDC__ should always be defined
 * to 1 for compilation modes that accept ANSI C syntax regardless of whether
 * or not extensions to the C standard are used. Violations of conforming
 * behavior are conditionally flagged as warnings via the use of the
 * -pedantic option. In addition to defining __STDC__ to 1, the GNU C
 * compiler also defines __STRICT_ANSI__ as a means of specifying strictly
 * conforming environments using the -ansi or -std=<standard> options.
 *
 * In the absence of any other compiler options, Sun and GNU set the value
 * of __STDC__ as follows when using the following options:
 *
 *                                  Value of __STDC__  __STRICT_ANSI__
 *
 * cc -Xa (default)                         0             undefined
 * cc -Xt (transitional)                    0             undefined
 * cc -Xc (strictly conforming)             1             undefined
 * cc -Xs (K&R C)                      undefined          undefined
 *
 * gcc (default)                            1             undefined
 * gcc -ansi, -std={c89, c99,...)           1              defined
 * gcc -traditional (K&R)              undefined          undefined
 *
 * The default compilation modes for Sun C compilers versus GNU C compilers
 * results in a differing value for __STDC__ which results in a more
 * restricted namespace when using Sun compilers. To allow both GNU and Sun
```

```
 * interpretations to peacefully co-exist, we use the following Sun
 * implementation _STRICT_STDC_ macro:
 */

#if (__STDC__ - 0 == 1 && !defined(__GNUC__)) || \
        (defined(__GNUC__) && defined(__STRICT_ANSI__))
#define _STRICT_STDC
#else
#undef  _STRICT_STDC
#endif

/*
 * Compiler complies with ISO/IEC 9899:1999
 */

#if __STDC_VERSION__ - 0 >= 199901L
#define _STDC_C99
#endif

/*
 * Large file interfaces:
 *
 *      _LARGEFILE_SOURCE
 *              1               large file-related additions to POSIX
 *                              interfaces requested (fseeko, etc.)
 *      _LARGEFILE64_SOURCE
 *              1               transitional large-file-related interfaces
 *                              requested (seek64, stat64, etc.)
 *
 * The corresponding announcement macros are respectively:
 *      _LFS_LARGEFILE
 *      _LFS64_LARGEFILE
 * (These are set in <unistd.h>.)
 *
 * Requesting _LARGEFILE64_SOURCE implies requesting _LARGEFILE_SOURCE as
 * well.
 *
 * The large file interfaces are made visible regardless of the initial values
 * of the feature test macros under certain circumstances:
 *    - If no explicit standards-conforming environment is requested (neither
 *      of _POSIX_SOURCE nor _XOPEN_SOURCE is defined and the value of
 *      __STDC__ does not imply standards conformance).
 *    - Extended system interfaces are explicitly requested (__EXTENSIONS__
 *      is defined).
 *    - Access to in-kernel interfaces is requested (_KERNEL or _KMEMUSER is
 *      defined).  (Note that this dependency is an artifact of the current
 *      kernel implementation and may change in future releases.)
 */
#if     (!defined(_STRICT_STDC) && !defined(__XOPEN_OR_POSIX)) || \
                defined(_KERNEL) || defined(_KMEMUSER) || \
                defined(__EXTENSIONS__)
#undef  _LARGEFILE64_SOURCE
#define _LARGEFILE64_SOURCE     1
#endif
#if     _LARGEFILE64_SOURCE - 0 == 1
#undef  _LARGEFILE_SOURCE
#define _LARGEFILE_SOURCE       1
#endif

/*
 * Large file compilation environment control:
 *
 * The setting of _FILE_OFFSET_BITS controls the size of various file-related
 * types and governs the mapping between file-related source function symbol
 * names and the corresponding binary entry points.
 *
```



```
 * In the 32-bit environment, the default value is 32; if not set, set it to
 * the default here, to simplify tests in other headers.
 *
 * In the 64-bit compilation environment, the only value allowed is 64.
 */
#if defined(_LP64)
#ifndef _FILE_OFFSET_BITS
#define _FILE_OFFSET_BITS       64
#endif
#if     _FILE_OFFSET_BITS - 0 != 64
#error  "invalid _FILE_OFFSET_BITS value specified"
#endif
#else   /* _LP64 */
#ifndef _FILE_OFFSET_BITS
#define _FILE_OFFSET_BITS       32
#endif
#if     _FILE_OFFSET_BITS - 0 != 32 && _FILE_OFFSET_BITS - 0 != 64
#error  "invalid _FILE_OFFSET_BITS value specified"
#endif
#endif  /* _LP64 */

/*
 * Use of _XOPEN_SOURCE
 *
 * The following X/Open specifications are supported:
 *
 * X/Open Portability Guide, Issue 3 (XPG3)
 * X/Open CAE Specification, Issue 4 (XPG4)
 * X/Open CAE Specification, Issue 4, Version 2 (XPG4v2)
 * X/Open CAE Specification, Issue 5 (XPG5)
 * Open Group Technical Standard, Issue 6 (XPG6), also referred to as
 *    IEEE Std. 1003.1-2001 and ISO/IEC 9945:2002.
 *
 * XPG4v2 is also referred to as UNIX 95 (SUS or SUSv1).
 * XPG5 is also referred to as UNIX 98 or the Single Unix Specification,
 *     Version 2 (SUSv2)
 * XPG6 is the result of a merge of the X/Open and POSIX specifications
 *     and as such is also referred to as IEEE Std. 1003.1-2001 in
 *     addition to UNIX 03 and SUSv3.
 *
 * When writing a conforming X/Open application, as per the specification
 * requirements, the appropriate feature test macros must be defined at
 * compile time. These are as follows. For more info, see standards(5).
 *
 * Feature Test Macro                                    Specification
 * ------------------------------------------------  -------------
 * _XOPEN_SOURCE >= 600  (or POSIX_C_SOURCE>=200112L)    XPG6
 * _XOPEN_SOURCE >= 500                                  XPG5
 * _XOPEN_SOURCE && _XOPEN_SOURCE_EXTENDED = 1           XPG4v2
 * _XOPEN_SOURCE && _XOPEN_VERSION = 4                   XPG4
#endif /* ! codereview */
 * _XOPEN_SOURCE                                         XPG3
 * _XOPEN_SOURCE && _XOPEN_VERSION = 4                   XPG4
 * _XOPEN_SOURCE && _XOPEN_SOURCE_EXTENDED = 1           XPG4v2
 * _XOPEN_SOURCE = 500                                   XPG5
 * _XOPEN_SOURCE = 600  (or POSIX_C_SOURCE=200112L)      XPG6
 *
 * In order to simplify the guards within the headers, the following
 * implementation private test macros have been created. Applications
 * must NOT use these private test macros as unexpected results will
 * occur.
 *
 * Note that in general, the use of these private macros is cumulative.
 * For example, the use of _XPG3 with no other restrictions on the X/Open
 * namespace will make the symbols visible for XPG3 through XPG6
 * compilation environments. The use of _XPG4_2 with no other X/Open
```

```
 255  * namespace restrictions indicates that the symbols were introduced in
 256  * XPG4v2 and are therefore visible for XPG4v2 through XPG6 compilation
 257  * environments, but not for XPG3 or XPG4 compilation environments.
 258  *
 259  * _XPG3    X/Open Portability Guide, Issue 3 (XPG3)
 260  * _XPG4    X/Open CAE Specification, Issue 4 (XPG4)
 261  * _XPG4_2  X/Open CAE Specification, Issue 4, Version 2 (XPG4v2/UNIX 95/SUS)
 262  * _XPG5    X/Open CAE Specification, Issue 5 (XPG5/UNIX 98/SUSv2)
 263  * _XPG6    Open Group Technical Standard, Issue 6 (XPG6/UNIX 03/SUSv3)
 264  */

 266 /* Open Group Technical Standard , Issue 6 */
 267 #if     (_XOPEN_SOURCE - 0 >= 600) || (_POSIX_C_SOURCE - 0 >= 200112L)
 268 #define _XPG6
 269 #define _XPG5
 264 /* X/Open Portability Guide, Issue 3 */
 265 #if defined(_XOPEN_SOURCE) && (_XOPEN_SOURCE - 0 < 500) && \
 266         (_XOPEN_VERSION - 0 < 4) && !defined(_XOPEN_SOURCE_EXTENDED)
 267 #define _XPG3
 268 /* X/Open CAE Specification, Issue 4 */
 269 #elif   (defined(_XOPEN_SOURCE) && _XOPEN_VERSION - 0 == 4)
 270 #define _XPG4
 271 #define _XPG3
 272 /* X/Open CAE Specification, Issue 4, Version 2 */
 273 #elif (defined(_XOPEN_SOURCE) && _XOPEN_SOURCE_EXTENDED - 0 == 1)
 270 #define _XPG4_2
 271 #define _XPG4
 272 #define _XPG3
 273 #undef  _POSIX_C_SOURCE
 274 #define _POSIX_C_SOURCE                 200112L
 275 #undef  _XOPEN_SOURCE
 276 #define _XOPEN_SOURCE                   600
 277 #endif /* ! codereview */
 278 /* X/Open CAE Specification, Issue 5 */
 279 #elif   (_XOPEN_SOURCE - 0 >= 500)
 277 #elif   (_XOPEN_SOURCE - 0 == 500)
 280 #define _XPG5
 281 #define _XPG4_2
 282 #define _XPG4
 283 #define _XPG3
 284 #undef  _POSIX_C_SOURCE
 285 #define _POSIX_C_SOURCE                 199506L
 286 /* X/Open CAE Specification, Issue 4, Version 2 */
 287 #elif (defined(_XOPEN_SOURCE) && _XOPEN_SOURCE_EXTENDED - 0 == 1)
 284 /* Open Group Technical Standard , Issue 6 */
 285 #elif   (_XOPEN_SOURCE - 0 == 600) || (_POSIX_C_SOURCE - 0 == 200112L)
 286 #define _XPG6
 287 #define _XPG5
 288 #define _XPG4_2
 289 #define _XPG4
 290 #define _XPG3
 291 /* X/Open CAE Specification, Issue 4 */
 292 #elif   (defined(_XOPEN_SOURCE) && _XOPEN_VERSION - 0 == 4)
 293 #define _XPG4
 294 #define _XPG3
 295 /* X/Open Portability Guide, Issue 3 */
 296 #elif defined(_XOPEN_SOURCE) && (_XOPEN_VERSION - 0 < 4)
 297 #define _XPG3
 291 #undef  _POSIX_C_SOURCE
 292 #define _POSIX_C_SOURCE                 200112L
 293 #undef  _XOPEN_SOURCE
 294 #define _XOPEN_SOURCE                   600
 298 #endif

 300 /*
 301  * _XOPEN_VERSION is defined by the X/Open specifications and is not
```



```
 302  * normally defined by the application, except in the case of an XPG4
 303  * application.  On the implementation side, _XOPEN_VERSION defined with
 304  * the value of 3 indicates an XPG3 application. _XOPEN_VERSION defined
 305  * with the value of 4 indicates an XPG4 or XPG4v2 (UNIX 95) application.
 306  * _XOPEN_VERSION  defined with a value of 500 indicates an XPG5 (UNIX 98)
 307  * application and with a value of 600 indicates an XPG6 (UNIX 03)
 308  * application.  The appropriate version is determined by the use of the
 309  * feature test macros described earlier.  The value of _XOPEN_VERSION
 310  * defaults to 3 otherwise indicating support for XPG3 applications.
 311  */
 312 #ifndef _XOPEN_VERSION
 313 #ifdef  _XPG6
 314 #define _XOPEN_VERSION 600
 315 #elif defined(_XPG5)
 316 #define _XOPEN_VERSION 500
 317 #elif   defined(_XPG4_2)
 318 #define _XOPEN_VERSION  4
 319 #else
 320 #define _XOPEN_VERSION  3
 321 #endif
 322 #endif

 324 /*
 325  * ANSI C and ISO 9899:1990 say the type long long doesn't exist in strictly
 326  * conforming environments.  ISO 9899:1999 says it does.
 327  *
 328  * The presence of _LONGLONG_TYPE says "long long exists" which is therefore
 329  * defined in all but strictly conforming environments that disallow it.
 330  */
 331 #if !defined(_STDC_C99) && defined(_STRICT_STDC) && !defined(__GNUC__)
 332 /*
 333  * Resist attempts to force the definition of long long in this case.
 334  */
 335 #if defined(_LONGLONG_TYPE)
 336 #error  "No long long in strictly conforming ANSI C & 1990 ISO C environments"
 337 #endif
 338 #else
 339 #if !defined(_LONGLONG_TYPE)
 340 #define _LONGLONG_TYPE
 341 #endif
 342 #endif

 344 /*
 345  * It is invalid to compile an XPG3, XPG4, XPG4v2, or XPG5 application
 346  * using c99.  The same is true for POSIX.1-1990, POSIX.2-1992, POSIX.1b,
 347  * and POSIX.1c applications. Likewise, it is invalid to compile an XPG6
 348  * or a POSIX.1-2001 application with anything other than a c99 or later
 349  * compiler.  Therefore, we force an error in both cases.
 350  */
 351 #if defined(_STDC_C99) && (defined(__XOPEN_OR_POSIX) && !defined(_XPG6))
 352 #error "Compiler or options invalid for pre-UNIX 03 X/Open applications \
 353         and pre-2001 POSIX applications"
 354 #elif !defined(_STDC_C99) && \
 355         (defined(__XOPEN_OR_POSIX) && defined(_XPG6))
 356 #error "Compiler or options invalid; UNIX 03 and POSIX.1-2001 applications \
 357         require the use of c99"
 358 #endif

 360 /*
 361  * The following macro defines a value for the ISO C99 restrict
 362  * keyword so that _RESTRICT_KYWD resolves to "restrict" if
 363  * an ISO C99 compiler is used and "" (null string) if any other
 364  * compiler is used. This allows for the use of single prototype
 365  * declarations regardless of compiler version.
 366  */
 367 #if (defined(__STDC__) && defined(_STDC_C99)) && !defined(__cplusplus)
```

```
#define _RESTRICT_KYWD  restrict
#else
#define _RESTRICT_KYWD
#endif

/*
 * The following macro indicates header support for the ANSI C++
 * standard.  The ISO/IEC designation for this is ISO/IEC FDIS 14882.
 */
#define _ISO_CPP_14882_1998

/*
 * The following macro indicates header support for the C99 standard,
 * ISO/IEC 9899:1999, Programming Languages - C.
 */
#define _ISO_C_9899_1999

/*
 * The following macro indicates header support for DTrace. The value is an
 * integer that corresponds to the major version number for DTrace.
 */
#define _DTRACE_VERSION 1

#ifdef  __cplusplus
}
```

unchanged_portion_omitted_