```
**********************************************************
    6989 Thu Aug 28 16:35:32 2008
new/usr/src/uts/common/sys/byteorder.h
6729208 Optimize BSWAP_* and BE_* macros in sys/byteorder.h to use inline amd64
**********************************************************
```

```
/*
 * 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 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*      Copyright (c) 1983, 1984, 1985, 1986, 1987, 1988, 1989 AT&T     */
/*        All Rights Reserved   */

/*
 * University Copyright- Copyright (c) 1982, 1986, 1988
 * The Regents of the University of California
 * All Rights Reserved
 *
 * University Acknowledgment- Portions of this document are derived from
 * software developed by the University of California, Berkeley, and its
 * contributors.
 */

#ifndef _SYS_BYTEORDER_H
#define _SYS_BYTEORDER_H

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

#if defined(__GNUC__) && defined(_ASM_INLINES) && \
        (defined(__i386) || defined(__amd64))
#include <asm/byteorder.h>
#endif

#ifdef  __cplusplus
extern "C" {
#endif

/*
 * macros for conversion between host and (internet) network byte order
 */

#if defined(_BIG_ENDIAN) && !defined(ntohl) && !defined(__lint)
/* big-endian */
#define ntohl(x)        (x)
```



```
#define ntohll(x)       (x)
#define ntohs(x)        (x)
#define htonl(x)        (x)
#define htonll(x)       (x)
#define htons(x)        (x)

#elif !defined(ntohl) /* little-endian */

#ifndef _IN_PORT_T
#define _IN_PORT_T
typedef uint16_t in_port_t;
#endif

#ifndef _IN_ADDR_T
#define _IN_ADDR_T
typedef uint32_t in_addr_t;
#endif

#if !defined(_XPG4_2) || defined(__EXTENSIONS__) || defined(_XPG5)
extern  uint32_t htonl(uint32_t);
extern  uint16_t htons(uint16_t);
extern  uint32_t ntohl(uint32_t);
extern  uint16_t ntohs(uint16_t);
#else
extern  in_addr_t htonl(in_addr_t);
extern  in_port_t htons(in_port_t);
extern  in_addr_t ntohl(in_addr_t);
extern  in_port_t ntohs(in_port_t);
#endif  /* !defined(_XPG4_2) || defined(__EXTENSIONS__) || defined(_XPG5) */
#if !(defined(_XPG4_2) || defined(_XPG5)) || defined(__EXTENSIONS__)
extern  uint64_t htonll(uint64_t);
extern  uint64_t ntohll(uint64_t);
#endif  /* !(_XPG4_2||_XPG5) || __EXTENSIONS__ */
#endif

#if !defined(_XPG4_2) || defined(__EXTENSIONS__)

/*
 * Macros to reverse byte order
 */
#define BSWAP_8(x)      ((x) & 0xff)
#if !defined(__i386) && !defined(__amd64)
#define BSWAP_16(x)     ((BSWAP_8(x) << 8) | BSWAP_8((x) >> 8))
#define BSWAP_32(x)     (((uint32_t)(x) << 24) | \
                         (((uint32_t)(x) << 8) & 0xff0000) | \
                         (((uint32_t)(x) >> 8) & 0xff00) | \
                         ((uint32_t)(x)  >> 24))
#else /* x86 */
#define BSWAP_16(x)     htons(x)
#define BSWAP_32(x)     htonl(x)
#endif  /* !__i386 && !__amd64 */
#define BSWAP_32(x)     ((BSWAP_16(x) << 16) | BSWAP_16((x) >> 16))
#define BSWAP_64(x)     ((BSWAP_32(x) << 32) | BSWAP_32((x) >> 32))

#if (!defined(__i386) && !defined(__amd64)) || \
        ((defined(_XPG4_2) || defined(_XPG5)) && !defined(__EXTENSIONS__))
#define BSWAP_64(x)     (((uint64_t)(x) << 56) | \
                         (((uint64_t)(x) << 40) & 0xff000000000000ULL) | \
                         (((uint64_t)(x) << 24) & 0xff0000000000ULL) | \
                         (((uint64_t)(x) << 8)  & 0xff00000000ULL) | \
                         (((uint64_t)(x) >> 8)  & 0xff000000ULL) | \
                         (((uint64_t)(x) >> 24) & 0xff0000ULL) | \
                         (((uint64_t)(x) >> 40) & 0xff00ULL) | \
                         ((uint64_t)(x)  >> 56))
#else /* x86 with non-XPG extensions allowed */
#define BSWAP_64(x)     htonll(x)
```

```
#endif  /* (!__i386&&!__amd64) || ((_XPG4_2||_XPG5) && !__EXTENSIONS__) */

#define BMASK_8(x)      ((x) & 0xff)
#define BMASK_16(x)     ((x) & 0xffff)
#define BMASK_32(x)     ((x) & 0xffffffff)
#define BMASK_64(x)     (x)

/*
 * Macros to convert from a specific byte order to/from native byte order
 */
#ifdef _BIG_ENDIAN
#define BE_8(x)         BMASK_8(x)
#define BE_16(x)        BMASK_16(x)
#define BE_32(x)        BMASK_32(x)
#define BE_64(x)        BMASK_64(x)
#define LE_8(x)         BSWAP_8(x)
#define LE_16(x)        BSWAP_16(x)
#define LE_32(x)        BSWAP_32(x)
#define LE_64(x)        BSWAP_64(x)
#else
#define LE_8(x)         BMASK_8(x)
#define LE_16(x)        BMASK_16(x)
#define LE_32(x)        BMASK_32(x)
#define LE_64(x)        BMASK_64(x)
#define BE_8(x)         BSWAP_8(x)
#define BE_16(x)        BSWAP_16(x)
#define BE_32(x)        BSWAP_32(x)
#define BE_64(x)        BSWAP_64(x)
#endif

/*
 * Macros to read unaligned values from a specific byte order to
 * native byte order
 */

#define BE_IN8(xa) \
        *((uint8_t *)(xa))

#if !defined(__i386) && !defined(__amd64)
#define BE_IN16(xa) \
        (((uint16_t)BE_IN8(xa) << 8) | BE_IN8((uint8_t *)(xa) + 1))
        (((uint16_t)BE_IN8(xa) << 8) | BE_IN8((uint8_t *)(xa)+1))

#define BE_IN32(xa) \
        (((uint32_t)BE_IN16(xa) << 16) | BE_IN16((uint8_t *)(xa) + 2))
        (((uint32_t)BE_IN16(xa) << 16) | BE_IN16((uint8_t *)(xa)+2))

#else /* x86 */
#define BE_IN16(xa) htons(*((uint16_t *)(void *)(xa)))
#define BE_IN32(xa) htonl(*((uint32_t *)(void *)(xa)))
#endif  /* !__i386 && !__amd64 */

#if (!defined(__i386) && !defined(__amd64)) || \
        ((defined(_XPG4_2) || defined(_XPG5)) && !defined(__EXTENSIONS__))
#define BE_IN64(xa) \
        (((uint64_t)BE_IN32(xa) << 32) | BE_IN32((uint8_t *)(xa) + 4))
#else /* x86 with non-XPG extensions allowed */
#define BE_IN64(xa) htonll(*((uint64_t *)(void *)(xa)))
#endif  /* (!__i386&&!__amd64) || ((_XPG4_2||_XPG5) && !__EXTENSIONS__) */
        (((uint64_t)BE_IN32(xa) << 32) | BE_IN32((uint8_t *)(xa)+4))

#define LE_IN8(xa) \
        *((uint8_t *)(xa))

#define LE_IN16(xa) \
        (((uint16_t)LE_IN8((uint8_t *)(xa) + 1) << 8) | LE_IN8(xa))
```



```
#define LE_IN32(xa) \
        (((uint32_t)LE_IN16((uint8_t *)(xa) + 2) << 16) | LE_IN16(xa))

#define LE_IN64(xa) \
        (((uint64_t)LE_IN32((uint8_t *)(xa) + 4) << 32) | LE_IN32(xa))

/*
 * Macros to write unaligned values from native byte order to a specific byte
 * order.
 */

#define BE_OUT8(xa, yv) *((uint8_t *)(xa)) = (uint8_t)(yv);

#define BE_OUT16(xa, yv) \
        BE_OUT8((uint8_t *)(xa) + 1, yv); \
        BE_OUT8((uint8_t *)(xa), (yv) >> 8);

#define BE_OUT32(xa, yv) \
        BE_OUT16((uint8_t *)(xa) + 2, yv); \
        BE_OUT16((uint8_t *)(xa), (yv) >> 16);

#if (!defined(__i386) && !defined(__amd64)) || \
        ((defined(_XPG4_2) || defined(_XPG5)) && !defined(__EXTENSIONS__))
#define BE_OUT64(xa, yv) \
        BE_OUT32((uint8_t *)(xa) + 4, yv); \
        BE_OUT32((uint8_t *)(xa), (yv) >> 32);
#else /* x86 with non-XPG extensions allowed */
#define BE_OUT64(xa, yv) *((uint64_t *)(void *)(xa)) = htonll((uint64_t)(yv));
#endif  /* (!__i386&&!__amd64) || ((_XPG4_2||_XPG5) && !__EXTENSIONS__) */

#define LE_OUT8(xa, yv) *((uint8_t *)(xa)) = (uint8_t)(yv);

#define LE_OUT16(xa, yv) \
        LE_OUT8((uint8_t *)(xa), yv); \
        LE_OUT8((uint8_t *)(xa) + 1, (yv) >> 8);

#define LE_OUT32(xa, yv) \
        LE_OUT16((uint8_t *)(xa), yv); \
        LE_OUT16((uint8_t *)(xa) + 2, (yv) >> 16);

#define LE_OUT64(xa, yv) \
        LE_OUT32((uint8_t *)(xa), yv); \
        LE_OUT32((uint8_t *)(xa) + 4, (yv) >> 32);

#endif  /* !defined(_XPG4_2) || defined(__EXTENSIONS__) */

#ifdef  __cplusplus
}
```

*unchanged_portion_omitted_*