```
**********************************************************
  161892 Wed Aug 13 12:15:18 2014
new/usr/src/cmd/fs.d/ufs/mkfs/mkfs.c
5083 avoid undefined order of operations in assignments
**********************************************************
______unchanged_portion_omitted_
#endif
char buf[MAXBSIZE];

static void
fsinit()
{
        int i;


        /*
         * initialize the node
         */
        node.i_atime = mkfstime;
        node.i_mtime = mkfstime;
        node.i_ctime = mkfstime;
#ifdef LOSTDIR
        /*
         * create the lost+found directory
         */
        (void) makedir(lost_found_dir, 2);
        for (i = DIRBLKSIZ; i < sblock.fs_bsize; i += DIRBLKSIZ) {
                bcopy(&lost_found_dir[2], &buf[i], DIRSIZ(&lost_found_dir[2]));
        }
        node.i_number = LOSTFOUNDINO;
        node.i_smode = IFDIR | 0700;
        node.i_smode = node.i_mode = IFDIR | 0700;
        node.i_nlink = 2;
        node.i_size = sblock.fs_bsize;
        node.i_db[0] = alloc((int)node.i_size, node.i_mode);
        node.i_blocks = btodb(fragroundup(&sblock, (int)node.i_size));
        IRANDOMIZE(&node.i_ic);
        wtfs(fsbtodb(&sblock, (uint64_t)node.i_db[0]), (int)node.i_size, buf);
        iput(&node);
#endif
        /*
         * create the root directory
         */
        node.i_number = UFSROOTINO;
        node.i_mode = IFDIR | UMASK;
        node.i_mode = node.i_smode = IFDIR | UMASK;
        node.i_nlink = PREDEFDIR;
        node.i_size = makedir(root_dir, PREDEFDIR);
        node.i_db[0] = alloc(sblock.fs_fsize, node.i_mode);
        /* i_size < 2GB because we are initializing the file system */
        node.i_blocks = btodb(fragroundup(&sblock, (int)node.i_size));
        IRANDOMIZE(&node.i_ic);
        wtfs(fsbtodb(&sblock, (uint64_t)node.i_db[0]), sblock.fs_fsize, buf);
        iput(&node);
}
______unchanged_portion_omitted_
```

```
**********************************************************
   18770 Wed Aug 13 12:15:19 2014
new/usr/src/cmd/mdb/intel/modules/generic_cpu/gcpu.c
5083 avoid undefined order of operations in assignments
**********************************************************
______unchanged_portion_omitted_

/*
 * Advance the <chipid,coreid,strandid> tuple to the next strand entry
 * Return true upon sucessful result. Otherwise return false if already reach
 * the highest strand.
 */
static boolean_t
cmih_ent_next(struct cmih_walk_state *wsp)
{
        uint_t carry = 0;

        /* Check for end of the table */
        if (wsp->chipid >= CMI_MAX_CHIPID &&
            wsp->coreid >= CMI_MAX_COREID(wsp->core_nbits) &&
            wsp->strandid >= CMI_MAX_STRANDID(wsp->strand_nbits))
                return (B_FALSE);

        /* increment the strand id */
        wsp->strandid++;
        carry =  wsp->strandid >> wsp->strand_nbits;
        wsp->strandid =  wsp->strandid & CMI_MAX_STRANDID(wsp->strand_nbits);
        if (carry == 0)
                return (B_TRUE);

        /* increment the core id */
        wsp->coreid++;
        carry = wsp->coreid >> wsp->core_nbits;
        wsp->coreid = wsp->coreid & CMI_MAX_COREID(wsp->core_nbits);
        if (carry == 0)
                return (B_TRUE);

        /* increment the chip id */
        wsp->chipid = (wsp->chipid + 1) & (CMI_MAX_CHIPID);
        wsp->chipid = ++wsp->chipid & (CMI_MAX_CHIPID);

        return (B_TRUE);
}
______unchanged_portion_omitted_
```

```
**********************************************************
    4004 Wed Aug 13 12:15:20 2014
new/usr/src/cmd/scadm/sparc/mpxu/common/smq.c
5083 avoid undefined order of operations in assignments
**********************************************************
```

```
/*
 * 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 2002 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * smq.c: to provide a message queue system for scadm functions (used in the
 * firmware download context where BP messages, received from the service
 * processor, are stored in the message queue)
 *
 * these routines come from the libxposix library
 */

#include <sys/types.h>
#include <time.h>

#include "xsem.h"
#include "smq.h"


#define SMQ_VALID_SMQ           0x0000003b
#define SMQ_VALID_SMQ_MASK      0x000000FF


int
smq_init(smq_t *smq, smq_msg_t *msgbuffer, int depth)
{
        /* allocate local semaphore initialized to 0 */
        if (xsem_init(&smq->smq_msgAvail, 0, 0) != 0)
                return (SMQ_ERROR);

        smq->smq_control        = SMQ_VALID_SMQ;
        smq->smq_msgBuffer      = msgbuffer;
        smq->smq_head           = msgbuffer;
        smq->smq_tail           = msgbuffer;
        smq->smq_count          = 0;
        smq->smq_depth          = depth;
```

```
        return (0);
}
```
______***unchanged_portion_omitted_***

```
int
smq_receive(smq_t *smq, smq_msg_t *msg)
{
        if ((smq->smq_control & SMQ_VALID_SMQ_MASK) != SMQ_VALID_SMQ)
                return (SMQ_INVALID);

        /* Wait for message */
        (void) xsem_wait(&smq->smq_msgAvail);

        if (smq->smq_count == 0)
                return (SMQ_ERROR);

        /* Copy messaged into queue */
        *msg = *smq->smq_head;

        smq->smq_head++;
        /* Increment Head */
        smq->smq_head = smq->smq_head++;
        if ((unsigned long)smq->smq_head > ((unsigned long)smq->smq_msgBuffer +
            (unsigned long)(smq->smq_depth * sizeof (smq_msg_t)))) {
                smq->smq_head = smq->smq_msgBuffer;
        }
        smq->smq_count--;

        return (0);
}


int
smq_send(smq_t *smq, smq_msg_t *msg)
{
        if ((smq->smq_control & SMQ_VALID_SMQ_MASK) != SMQ_VALID_SMQ)
                return (SMQ_INVALID);

        if (smq->smq_count == smq->smq_depth)
                return (SMQ_FULL);

        /* Copy messaged into queue */
        *smq->smq_tail = *msg;

        smq->smq_tail++;
        /* Increment Tail */
        smq->smq_tail = smq->smq_tail++;
        if ((unsigned long)smq->smq_tail > ((unsigned long)smq->smq_msgBuffer +
            (unsigned long)(smq->smq_depth * sizeof (smq_msg_t)))) {
                smq->smq_tail = smq->smq_msgBuffer;
        }

        smq->smq_count++;
        (void) xsem_post(&smq->smq_msgAvail);

        return (0);
}
```
______***unchanged_portion_omitted_***

```
int
smq_xreceive(smq_t *smq, timestruc_t *timeout, smq_msg_t *msg)
{
        int Status;
        if ((smq->smq_control & SMQ_VALID_SMQ_MASK) != SMQ_VALID_SMQ)
                return (SMQ_INVALID);

        /* Wait for message */
        if ((Status = xsem_xwait(&smq->smq_msgAvail, 1, timeout)) == XSEM_ETIME)
                return (SMQ_ETIME);

        if (Status != 0)
                return (SMQ_ERROR);

        if (smq->smq_count == 0)
                return (SMQ_ERROR);

        /* Copy messaged into queue */
        *msg = *smq->smq_head;

        smq->smq_head++;
        /* Increment Head */
        smq->smq_head = smq->smq_head++;
        if ((unsigned long)smq->smq_head > ((unsigned long)smq->smq_msgBuffer +
            (unsigned long)(smq->smq_depth * sizeof (smq_msg_t)))) {
                smq->smq_head = smq->smq_msgBuffer;
        }
        smq->smq_count--;


        return (0);
}
```
______***unchanged_portion_omitted_***

```
**********************************************************
    7083 Wed Aug 13 12:15:21 2014
new/usr/src/cmd/troff/nroff.d/n6.c
5083 avoid undefined order of operations in assignments
**********************************************************
/*
 * 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 2003 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*      Copyright (c) 1984, 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.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

#include "tdef.h"
#include "tw.h"
#include "ext.h"
#include <ctype.h>

/*
 * n6.c -- width functions, sizes and fonts
*/

int     bdtab[NFONT+1] ={ 0, 0, 0, 3, 3, 0, };
int     sbold = 0;
int     fontlab[NFONT+1] = { 0, 'R', 'I', 'B', PAIR('B','I'), 'S', 0 };

extern  int     nchtab;

int
width(j)
tchar j;
{
        int     i, k;
```

```
        if (j & (ZBIT|MOT)) {
                if (iszbit(j))
                        return(0);
                if (isvmot(j))
                        return(0);
                k = absmot(j);
                if (isnmot(j))
                        k = -k;
                return(k);
        }
        i = cbits(j);
        if (i < ' ') {
                if (i == '\b')
                        return(-widthp);
                if (i == PRESC)
                        i = eschar;
                else if (iscontrol(i))
                        return(0);
        }
        if (i==ohc)
                return(0);
#ifdef EUC
#ifdef NROFF
        if (multi_locale) {
                if ((j & MBMASK) || (j & CSMASK)) {
                        switch(j & MBMASK) {
                                case BYTE_CHR:
                                case LASTOFMB:
                                        k = t.Char * csi_width[cs(j)];
                                        break;
                                default:
                                        k = 0;
                                        break;
                        }
                        widthp = k;
                        return(k);
                }
        }
        i &= 0x1ff;
#endif /* NROFF */
#endif /* EUC */
        i = trtab[i];
        if (i < 32)
                return(0);
        k = t.width[i] * t.Char;
        widthp = k;
        return(k);
}
______unchanged_portion_omitted_


int
setwd()
{
        int     base, wid;
        tchar i;
        int     delim, emsz, k;
        int     savhp, savapts, savapts1, savfont, savfont1, savpts, savpts1;

        base = numtab[ST].val = wid = numtab[CT].val = 0;
        base = numtab[ST].val = numtab[ST].val = wid = numtab[CT].val = 0;
        if (ismot(i = getch()))
                return (0);
        delim = cbits(i);
        savhp = numtab[HP].val;
```

```
	numtab[HP].val = 0;
	savapts = apts;
	savapts1 = apts1;
	savfont = font;
	savfont1 = font1;
	savpts = pts;
	savpts1 = pts1;
	setwdf++;
	while (cbits(i = getch()) != delim && !nlflg) {
		k = width(i);
		wid += k;
		numtab[HP].val += k;
		if (!ismot(i)) {
			emsz = (INCH * pts + 36) / 72;
		} else if (isvmot(i)) {
			k = absmot(i);
			if (isnmot(i))
				k = -k;
			base -= k;
			emsz = 0;
		} else
			continue;
		if (base < numtab[SB].val)
			numtab[SB].val = base;
		if ((k = base + emsz) > numtab[ST].val)
			numtab[ST].val = k;
	}
	setn1(wid, 0, (tchar) 0);
	numtab[HP].val = savhp;
	apts = savapts;
	apts1 = savapts1;
	font = savfont;
	font1 = savfont1;
	pts = savpts;
	pts1 = savpts1;
	mchbits();
	setwdf = 0;

	return (0);
}
```
______***unchanged_portion_omitted_***

```
**********************************************************
   14079 Wed Aug 13 12:15:21 2014
new/usr/src/cmd/troff/troff.d/t6.c
5083 avoid undefined order of operations in assignments
**********************************************************
/*
 * 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 2003 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*      Copyright (c) 1984, 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.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

/*
 * t6.c
 *
 * width functions, sizes and fonts
 */

#include "tdef.h"
#include "dev.h"
#include <ctype.h>
#include "ext.h"

/* fitab[f][c] is 0 if c is not on font f */
        /* if it's non-zero, c is in fontab[f] at position
         * fitab[f][c].
         */
extern  struct Font *fontbase[NFONT+1];
extern  char *codetab[NFONT+1];
extern int nchtab;

int     fontlab[NFONT+1];
```

```
short   *pstab;
int     cstab[NFONT+1];
int     ccstab[NFONT+1];
int     bdtab[NFONT+1];
int     sbold = 0;

int
width(j)
tchar j;
{
        int     i, k;

        if (j & (ZBIT|MOT)) {
                if (iszbit(j))
                        return(0);
                if (isvmot(j))
                        return(0);
                k = absmot(j);
                if (isnmot(j))
                        k = -k;
                return(k);
        }
        i = cbits(j);
        if (i < ' ') {
                if (i == '\b')
                        return(-widthp);
                if (i == PRESC)
                        i = eschar;
                else if (iscontrol(i))
                        return(0);
        }
        if (i==ohc)
                return(0);
        i = trtab[i];
        if (i < 32)
                return(0);
        if (sfbits(j) == oldbits) {
                xfont = pfont;
                xpts = ppts;
        } else
                xbits(j, 0);
        if (widcache[i-32].fontpts == (xfont<<8) + xpts && !setwdf)
                k = widcache[i-32].width;
        else {
                k = getcw(i-32);
                if (bd)
                        k += (bd - 1) * HOR;
                if (cs)
                        k = cs;
        }
        widthp = k;
        return(k);
}
______unchanged_portion_omitted_


int
setwd()
{
        int     base, wid;
        tchar i;
        int     delim, emsz, k;
        int     savhp, savapts, savapts1, savfont, savfont1, savpts, savpts1;

        base = numtab[ST].val = wid = numtab[CT].val = 0;
        base = numtab[ST].val = numtab[ST].val = wid = numtab[CT].val = 0;
```

```
	if (ismot(i = getch()))
		return (0);
	delim = cbits(i);
	savhp = numtab[HP].val;
	numtab[HP].val = 0;
	savapts = apts;
	savapts1 = apts1;
	savfont = font;
	savfont1 = font1;
	savpts = pts;
	savpts1 = pts1;
	setwdf++;
	while (cbits(i = getch()) != delim && !nlflg) {
		k = width(i);
		wid += k;
		numtab[HP].val += k;
		if (!ismot(i)) {
			emsz = POINT * xpts;
		} else if (isvmot(i)) {
			k = absmot(i);
			if (isnmot(i))
				k = -k;
			base -= k;
			emsz = 0;
		} else
			continue;
		if (base < numtab[SB].val)
			numtab[SB].val = base;
		if ((k = base + emsz) > numtab[ST].val)
			numtab[ST].val = k;
	}
	setn1(wid, 0, (tchar) 0);
	numtab[HP].val = savhp;
	apts = savapts;
	apts1 = savapts1;
	font = savfont;
	font1 = savfont1;
	pts = savpts;
	pts1 = savpts1;
	mchbits();
	setwdf = 0;

	return (0);
}
```
______***unchanged_portion_omitted_***

*****************************************************************

***26392 Wed Aug 13 12:15:22 2014***
***new/usr/src/uts/common/io/hxge/hxge_virtual.c***
***5083 avoid undefined order of operations in assignments***

*****************************************************************

***______unchanged_portion_omitted_***

```
static void
hxge_ldgv_setup(p_hxge_ldg_t *ldgp, p_hxge_ldv_t *ldvp, uint8_t ldv,
        uint8_t endldg, int *ngrps)
{
        HXGE_DEBUG_MSG((NULL, INT_CTL, "==> hxge_ldgv_setup"));
        /* Assign the group number for each device. */
        (*ldvp)->ldg_assigned = (*ldgp)->ldg;
        (*ldvp)->ldgp = *ldgp;
        (*ldvp)->ldv = ldv;

        HXGE_DEBUG_MSG((NULL, INT_CTL,
            "==> hxge_ldgv_setup: ldv %d endldg %d ldg %d, ldvp $%p",
            ldv, endldg, (*ldgp)->ldg, (*ldgp)->ldvp));

        (*ldgp)->nldvs++;
        if ((*ldgp)->ldg == (endldg - 1)) {
                if ((*ldgp)->ldvp == NULL) {
                        (*ldgp)->ldvp = *ldvp;
                        *ngrps += 1;
                        HXGE_DEBUG_MSG((NULL, INT_CTL,
                            "==> hxge_ldgv_setup: ngrps %d", *ngrps));
                }
                HXGE_DEBUG_MSG((NULL, INT_CTL,
                    "==> hxge_ldgv_setup: ldvp $%p ngrps %d",
                    *ldvp, *ngrps));
                ++*ldvp;
        } else {
                (*ldgp)->ldvp = *ldvp;
                *ngrps += 1;
                HXGE_DEBUG_MSG((NULL, INT_CTL, "==> hxge_ldgv_setup(done): "
                    "ldv %d endldg %d ldg %d, ldvp $%p",
                    ldv, endldg, (*ldgp)->ldg, (*ldgp)->ldvp));
                ++*ldvp;
                ++*ldgp;
                (*ldvp) = ++*ldvp;
                (*ldgp) = ++*ldgp;
                HXGE_DEBUG_MSG((NULL, INT_CTL,
                    "==> hxge_ldgv_setup: new ngrps %d", *ngrps));
        }

        HXGE_DEBUG_MSG((NULL, INT_CTL, "==> hxge_ldgv_setup: "
            "ldg %d nldvs %d ldv %d ldvp $%p endldg %d ngrps %d",
            (*ldgp)->ldg, (*ldgp)->nldvs, ldv, ldvp, endldg, *ngrps));

        HXGE_DEBUG_MSG((NULL, INT_CTL, "<== hxge_ldgv_setup"));
}
```

***______unchanged_portion_omitted_***

***********************************************************
  107808 Wed Aug 13 12:15:22 2014
new/usr/src/uts/common/io/ib/adapters/hermon/hermon_cmd.c
5083 avoid undefined order of operations in assignments
***********************************************************
______unchanged_portion_omitted_

```
/*
 * hermon_impl_mbox_free()
 *    Context: Can be called from interrupt or base context.
 */
static void
hermon_impl_mbox_free(hermon_mboxlist_t *mblist, hermon_mbox_t **mb)
{
        uint_t          mbox_indx;

        mutex_enter(&mblist->mbl_lock);

        /* Pull the "index" from mailbox entry */
        mbox_indx = (*mb)->mb_indx;

        /*
         * If mailbox list is not empty, then insert the entry.  Otherwise,
         * this is the only entry.  So update the pointers appropriately.
         */
        if (mblist->mbl_entries_free++ != 0) {
                /* Update the current mailbox */
                (*mb)->mb_next = mblist->mbl_head_indx;
                (*mb)->mb_prev = mblist->mbl_tail_indx;

                /* Update head and tail mailboxes */
                mblist->mbl_mbox[mblist->mbl_head_indx].mb_prev = mbox_indx;
                mblist->mbl_mbox[mblist->mbl_tail_indx].mb_next = mbox_indx;

                /* Update tail index */
                mblist->mbl_tail_indx = mbox_indx;

        } else {
                /* Update the current mailbox */
                (*mb)->mb_next = mbox_indx;
                (*mb)->mb_prev = mbox_indx;

                /* Update head and tail indexes */
                mblist->mbl_tail_indx = mbox_indx;
                mblist->mbl_head_indx = mbox_indx;
        }

        /*
         * Because we can have both waiters (SLEEP treads waiting for a
         * cv_signal to continue processing) and pollers (NOSLEEP treads
         * polling for a mailbox to become available), we try to share CPU time
         * between them.  We do this by signalling the waiters only every other
         * call to mbox_free.  This gives the pollers a chance to get some CPU
         * time to do their command.  If we signalled every time, the pollers
         * would have a much harder time getting CPU time.
         *
         * If there are waiters and no pollers, then we signal always.
         *
         * Otherwise, if there are either no waiters, there may in fact be
         * pollers, so we do not signal in that case.
         */
        if (mblist->mbl_pollers > 0 && mblist->mbl_waiters > 0) {
                /* flip the signal value */
                mblist->mbl_signal = (mblist->mbl_signal + 1) % 2;
                mblist->mbl_signal = (++mblist->mbl_signal) % 2;
```

```
        } else if (mblist->mbl_waiters > 0) {
                mblist->mbl_signal = 1;
        } else {
                mblist->mbl_signal = 0;
        }

        /*
         * Depending on the conditions in the previous check, we signal only if
         * we are supposed to.
         */
        if (mblist->mbl_signal) {
                mblist->mbl_waiters--;
                cv_signal(&mblist->mbl_cv);
        }

        /* Clear out the mailbox entry pointer */
        *mb = NULL;

        mutex_exit(&mblist->mbl_lock);
}
```

______unchanged_portion_omitted_

***********************************************************
***90313 Wed Aug 13 12:15:23 2014***
***new/usr/src/uts/common/io/ib/adapters/tavor/tavor_cmd.c***
***5083 avoid undefined order of operations in assignments***
***********************************************************
______***unchanged_portion_omitted_***

```
/*
 * tavor_impl_mbox_free()
 *    Context: Can be called from interrupt or base context.
 */
static void
tavor_impl_mbox_free(tavor_mboxlist_t *mblist, tavor_mbox_t **mb)
{
        uint_t          mbox_indx;

        TAVOR_TNF_ENTER(tavor_impl_mbox_free);

        mutex_enter(&mblist->mbl_lock);

        /* Pull the "index" from mailbox entry */
        mbox_indx = (*mb)->mb_indx;

        /*
         * If mailbox list is not empty, then insert the entry.  Otherwise,
         * this is the only entry.  So update the pointers appropriately.
         */
        if (mblist->mbl_entries_free++ != 0) {
                /* Update the current mailbox */
                (*mb)->mb_next = mblist->mbl_head_indx;
                (*mb)->mb_prev = mblist->mbl_tail_indx;

                /* Update head and tail mailboxes */
                mblist->mbl_mbox[mblist->mbl_head_indx].mb_prev = mbox_indx;
                mblist->mbl_mbox[mblist->mbl_tail_indx].mb_next = mbox_indx;

                /* Update tail index */
                mblist->mbl_tail_indx = mbox_indx;

        } else {
                /* Update the current mailbox */
                (*mb)->mb_next = mbox_indx;
                (*mb)->mb_prev = mbox_indx;

                /* Update head and tail indexes */
                mblist->mbl_tail_indx = mbox_indx;
                mblist->mbl_head_indx = mbox_indx;
        }

        /*
         * Because we can have both waiters (SLEEP treads waiting for a
         * cv_signal to continue processing) and pollers (NOSLEEP treads
         * polling for a mailbox to become available), we try to share CPU time
         * between them.  We do this by signalling the waiters only every other
         * call to mbox_free.  This gives the pollers a chance to get some CPU
         * time to do their command.  If we signalled every time, the pollers
         * would have a much harder time getting CPU time.
         *
         * If there are waiters and no pollers, then we signal always.
         *
         * Otherwise, if there are either no waiters, there may in fact be
         * pollers, so we do not signal in that case.
         */
        if (mblist->mbl_pollers > 0 && mblist->mbl_waiters > 0) {
                /* flip the signal value */
```

```
                mblist->mbl_signal = (mblist->mbl_signal + 1) % 2;
                mblist->mbl_signal = (++mblist->mbl_signal) % 2;
        } else if (mblist->mbl_waiters > 0) {
                mblist->mbl_signal = 1;
        } else {
                mblist->mbl_signal = 0;
        }

        /*
         * Depending on the conditions in the previous check, we signal only if
         * we are supposed to.
         */
        if (mblist->mbl_signal) {
                mblist->mbl_waiters--;
                cv_signal(&mblist->mbl_cv);
        }

        /* Clear out the mailbox entry pointer */
        *mb = NULL;

        mutex_exit(&mblist->mbl_lock);

        TAVOR_TNF_EXIT(tavor_impl_mbox_free);
}
```
______***unchanged_portion_omitted_***

```
**********************************************************
  115957 Wed Aug 13 12:15:23 2014
new/usr/src/uts/common/io/nxge/nxge_virtual.c
5083 avoid undefined order of operations in assignments
**********************************************************
______unchanged_portion_omitted_

/*
 * Static functions start here.
 */

static void
nxge_ldgv_setup(p_nxge_ldg_t *ldgp, p_nxge_ldv_t *ldvp, uint8_t ldv,
	uint8_t endldg, int *ngrps)
{
	NXGE_DEBUG_MSG((NULL, INT_CTL, "==> nxge_ldgv_setup"));
	/* Assign the group number for each device. */
	(*ldvp)->ldg_assigned = (*ldgp)->ldg;
	(*ldvp)->ldgp = *ldgp;
	(*ldvp)->ldv = ldv;

	NXGE_DEBUG_MSG((NULL, INT_CTL, "==> nxge_ldgv_setup: "
	    "ldv %d endldg %d ldg %d, ldvp $%p",
	    ldv, endldg, (*ldgp)->ldg, (*ldgp)->ldvp));

	(*ldgp)->nldvs++;
	if ((*ldgp)->ldg == (endldg - 1)) {
		if ((*ldgp)->ldvp == NULL) {
			(*ldgp)->ldvp = *ldvp;
			*ngrps += 1;
			NXGE_DEBUG_MSG((NULL, INT_CTL,
			    "==> nxge_ldgv_setup: ngrps %d", *ngrps));
		}
		NXGE_DEBUG_MSG((NULL, INT_CTL,
		    "==> nxge_ldgv_setup: ldvp $%p ngrps %d",
		    *ldvp, *ngrps));
		++*ldvp;
	} else {
		(*ldgp)->ldvp = *ldvp;
		*ngrps += 1;
		NXGE_DEBUG_MSG((NULL, INT_CTL, "==> nxge_ldgv_setup(done): "
		    "ldv %d endldg %d ldg %d, ldvp $%p",
		    ldv, endldg, (*ldgp)->ldg, (*ldgp)->ldvp));
		++*ldvp;
		++*ldgp;
		(*ldvp) = ++*ldvp;
		(*ldgp) = ++*ldgp;
		NXGE_DEBUG_MSG((NULL, INT_CTL,
		    "==> nxge_ldgv_setup: new ngrps %d", *ngrps));
	}

	NXGE_DEBUG_MSG((NULL, INT_CTL, "==> nxge_ldgv_setup: "
	    "ldv %d ldvp $%p endldg %d ngrps %d",
	    ldv, ldvp, endldg, *ngrps));

	NXGE_DEBUG_MSG((NULL, INT_CTL, "<== nxge_ldgv_setup"));
}
______unchanged_portion_omitted_
```

```
**********************************************************
   32324 Wed Aug 13 12:15:24 2014
new/usr/src/uts/intel/io/dktp/drvobj/strategy.c
5083 avoid undefined order of operations in assignments
**********************************************************
______unchanged_portion_omitted_

/* fields in diskhd */
#define hd_cnt                  b_back
#define hd_private              b_forw
#define hd_flags                b_flags
#define hd_sync_next            av_forw
#define hd_async_next           av_back

#define hd_sync2async           sync_async_ratio

#define QNEAR_FORWARD           0x01
#define QNEAR_BACKWARD          0x02
#define QNEAR_ASYNCONLY         0x04
#define QNEAR_ASYNCALSO         0x08

#define DBLK(bp) ((unsigned long)(bp)->b_private)

#define BP_LT_BP(a, b) (DBLK(a) < DBLK(b))
#define BP_GT_BP(a, b) (DBLK(a) > DBLK(b))
#define BP_LT_HD(a, b) (DBLK(a) < (unsigned long)((b)->hd_private))
#define BP_GT_HD(a, b) (DBLK(a) > (unsigned long)((b)->hd_private))
#define QNEAR_ASYNC     (QNEAR_ASYNCONLY|QNEAR_ASYNCALSO)

#define SYNC2ASYNC(a) ((a)->q_tab.hd_cnt)


/*
 * qmerge implements a two priority queue, the low priority queue holding ASYNC
 * write requests, while the rest are queued in the high priority sync queue.
 * Requests on the async queue would be merged if possible.
 * By default qmerge2wayscan is 1, indicating an elevator algorithm. When
 * this variable is set to zero, it has the following side effects.
 * 1. We assume fairness is the number one issue.
 * 2. The next request to be picked indicates current head position.
 *
 * qmerge_sync2async indicates the ratio of scans of high prioriy
 * sync queue to low priority async queue.
 *
 * When qmerge variables have the following values it defaults to qsort
 *
 * qmerge1pri = 1, qmerge2wayscan = 0, qmerge_max_merge = 0
 *
 */
static int      qmerge_max_merge = 128 * 1024;
static intptr_t qmerge_sync2async = 4;
static int      qmerge2wayscan = 1;
static int      qmerge1pri = 0;
static int      qmerge_merge = 0;

/*
 *      Local static data
 */
struct que_obj *
qmerge_create()
{
        struct que_data *qfp;
        struct que_obj *queobjp;

        queobjp = kmem_zalloc((sizeof (*queobjp) + sizeof (*qfp)), KM_NOSLEEP);
        if (!queobjp)
```

```
                return (NULL);

        queobjp->que_ops = &qmerge_ops;
        qfp = (struct que_data *)(queobjp+1);
        qfp->q_tab.hd_private = 0;
        qfp->q_tab.hd_private = qfp->q_tab.hd_private = 0;
        qfp->q_tab.hd_sync_next = qfp->q_tab.hd_async_next = NULL;
        qfp->q_tab.hd_cnt = (void *)qmerge_sync2async;
        queobjp->que_data = (opaque_t)qfp;

        return ((opaque_t)queobjp);
}
______unchanged_portion_omitted_
```

```
**********************************************************
   48527 Wed Aug 13 12:15:24 2014
new/usr/src/uts/sun4u/serengeti/io/sghsc.c
5083 avoid undefined order of operations in assignments
**********************************************************
______unchanged_portion_omitted_

/*
 * sghsc_rb_put()
 *      Insert an event info into the event ring buffer.
 * Returns DDI_FAILURE if the buffer is full, DDI_SUCCESS otherwise
 */
static int
sghsc_rb_put(sghsc_rb_head_t *rb_head, sghsc_event_t *event)
{
        if (rb_head->state == SGHSC_RB_FULL)
                return (DDI_FAILURE);

        rb_head->buf[rb_head->put_idx] = *event;

        rb_head->put_idx = (rb_head->put_idx + 1) & (rb_head->size - 1);
        rb_head->put_idx = ++rb_head->put_idx & (rb_head->size - 1);

        if (rb_head->put_idx == rb_head->get_idx)
                rb_head->state = SGHSC_RB_FULL;
        else
                rb_head->state = SGHSC_RB_FLOAT;

        return (DDI_SUCCESS);
}
/*
 * sghsc_rb_get()
 *      Remove an event info from the event  ring buffer.
 * Returns DDI_FAILURE if the buffer is empty, DDI_SUCCESS otherwise.
 */
static int
sghsc_rb_get(sghsc_rb_head_t *rb_head, sghsc_event_t *event)
{

        if (rb_head->state == SGHSC_RB_EMPTY)
                return (DDI_FAILURE);

        *event = rb_head->buf[rb_head->get_idx];

        rb_head->get_idx = (rb_head->get_idx + 1) & (rb_head->size - 1);
        rb_head->get_idx = ++rb_head->get_idx & (rb_head->size - 1);

        if (rb_head->get_idx == rb_head->put_idx)
                rb_head->state = SGHSC_RB_EMPTY;
        else
                rb_head->state = SGHSC_RB_FLOAT;

        return (DDI_SUCCESS);
}
______unchanged_portion_omitted_
```