

```
*****
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_
2789 #endif
2790 char buf[MAXBSIZE];

2792 static void
2793 fsinit()
2794 {
2795     int i;

2798     /*
2799     * initialize the node
2800     */
2801     node.i_atime = mkfstime;
2802     node.i_mtime = mkfstime;
2803     node.i_ctime = mkfstime;
2804 #ifdef LOSTDIR
2805     /*
2806     * create the lost+found directory
2807     */
2808     (void) mkdir(lost_found_dir, 2);
2809     for (i = DIRBLKSIZ; i < sblock.fs_bsize; i += DIRBLKSIZ) {
2810         bcopy(&lost_found_dir[2], &buf[i], DIRSIZ(&lost_found_dir[2]));
2811     }
2812     node.i_number = LOSTFOUNDINO;
2813     node.i_smode = IFDIR | 0700;
2813     node.i_smode = node.i_mode = IFDIR | 0700;
2814     node.i_nlink = 2;
2815     node.i_size = sblock.fs_bsize;
2816     node.i_db[0] = alloc((int)node.i_size, node.i_mode);
2817     node.i_blocks = btodb(fragroundup(&sblock, (int)node.i_size));
2818     IRANDOMIZE(&node.i_ic);
2819     wtfs(fsbtodb(&sblock, (uint64_t)node.i_db[0]), (int)node.i_size, buf);
2820     iput(&node);
2821 #endif

2822     /*
2823     * create the root directory
2824     */
2825     node.i_number = UFSROOTINO;
2826     node.i_mode = IFDIR | UMASK;
2826     node.i_mode = node.i_smode = IFDIR | UMASK;
2827     node.i_nlink = PREDEFDIR;
2828     node.i_size = mkdir(root_dir, PREDEFDIR);
2829     node.i_db[0] = alloc(sblock.fs_fsize, node.i_mode);
2830     /* i_size < 2GB because we are initializing the file system */
2831     node.i_blocks = btodb(fragroundup(&sblock, (int)node.i_size));
2832     IRANDOMIZE(&node.i_ic);
2833     wtfs(fsbtodb(&sblock, (uint64_t)node.i_db[0]), sblock.fs_fsize, buf);
2834     iput(&node);
2835 }
_____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

```
99 /*
100  * Advance the <chipid,coreid,strandid> tuple to the next strand entry
101  * Return true upon successful result. Otherwise return false if already reach
102  * the highest strand.
103  */
104 static boolean_t
105 cmih_ent_next(struct cmih_walk_state *wsp)
106 {
107     uint_t carry = 0;
108
109     /* Check for end of the table */
110     if (wsp->chipid >= CMI_MAX_CHIPID &&
111         wsp->coreid >= CMI_MAX_COREID(wsp->core_nbits) &&
112         wsp->strandid >= CMI_MAX_STRANDID(wsp->strand_nbits))
113         return (B_FALSE);
114
115     /* increment the strand id */
116     wsp->strandid++;
117     carry = wsp->strandid >> wsp->strand_nbits;
118     wsp->strandid = wsp->strandid & CMI_MAX_STRANDID(wsp->strand_nbits);
119     if (carry == 0)
120         return (B_TRUE);
121
122     /* increment the core id */
123     wsp->coreid++;
124     carry = wsp->coreid >> wsp->core_nbits;
125     wsp->coreid = wsp->coreid & CMI_MAX_COREID(wsp->core_nbits);
126     if (carry == 0)
127         return (B_TRUE);
128
129     /* increment the chip id */
130     wsp->chipid = (wsp->chipid + 1) & (CMI_MAX_CHIPID);
131     wsp->chipid = ++wsp->chipid & (CMI_MAX_CHIPID);
132
133     return (B_TRUE);
134 }
```

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
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License, Version 1.0 only
6  * (the "License"). You may not use this file except in compliance
7  * with the License.
8  *
9  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
10 * or http://www.opensolaris.org/os/licensing.
11 * See the License for the specific language governing permissions
12 * and limitations under the License.
13 *
14 * When distributing Covered Code, include this CDDL HEADER in each
15 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
16 * If applicable, add the following below this CDDL HEADER, with the
17 * fields enclosed by brackets "[]" replaced with your own identifying
18 * information: Portions Copyright [yyyy] [name of copyright owner]
19 *
20 * CDDL HEADER END
21 */
22 /*
23 * Copyright 2002 Sun Microsystems, Inc. All rights reserved.
24 * Use is subject to license terms.
25 */

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

28 /*
29 * smq.c: to provide a message queue system for scadm functions (used in the
30 * firmware download context where BP messages, received from the service
31 * processor, are stored in the message queue)
32 * these routines come from the libxposix library
33 */

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

38 #include "xsem.h"
39 #include "smq.h"

42 #define SMQ_VALID_SMQ 0x0000003b
43 #define SMQ_VALID_SMQ_MASK 0x000000FF

46 int
47 smq_init(smq_t *smq, smq_msg_t *msgbuffer, int depth)
48 {
49     /* allocate local semaphore initialized to 0 */
50     if (xsem_init(&smq->smq_msgAvail, 0, 0) != 0)
51         return (SMQ_ERROR);

53     smq->smq_control = SMQ_VALID_SMQ;
54     smq->smq_msgBuffer = msgbuffer;
55     smq->smq_head = msgbuffer;
56     smq->smq_tail = msgbuffer;
57     smq->smq_count = 0;
58     smq->smq_depth = depth;

```

```

60     return (0);
61 }
_____ unchanged_portion_omitted_

77 int
78 smq_receive(smq_t *smq, smq_msg_t *msg)
79 {
80     if ((smq->smq_control & SMQ_VALID_SMQ_MASK) != SMQ_VALID_SMQ)
81         return (SMQ_INVALID);

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

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

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

92     smq->smq_head++;
93     /* Increment Head */
94     smq->smq_head = smq->smq_head++;
95     if ((unsigned long)smq->smq_head > ((unsigned long)smq->smq_msgBuffer +
96         (unsigned long)(smq->smq_depth * sizeof (smq_msg_t)))) {
97         smq->smq_head = smq->smq_msgBuffer;
98     }
99     smq->smq_count--;
100 }

103 int
104 smq_send(smq_t *smq, smq_msg_t *msg)
105 {
106     if ((smq->smq_control & SMQ_VALID_SMQ_MASK) != SMQ_VALID_SMQ)
107         return (SMQ_INVALID);

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

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

115     smq->smq_tail++;
116     /* Increment Tail */
117     smq->smq_tail = smq->smq_tail++;
118     if ((unsigned long)smq->smq_tail > ((unsigned long)smq->smq_msgBuffer +
119         (unsigned long)(smq->smq_depth * sizeof (smq_msg_t)))) {
120         smq->smq_tail = smq->smq_msgBuffer;
121     }
122     smq->smq_count++;
123     (void) xsem_post(&smq->smq_msgAvail);

124     return (0);
125 }
_____ unchanged_portion_omitted_

152 int
153 smq_xreceive(smq_t *smq, timestruc_t *timeout, smq_msg_t *msg)
154 {
155     int Status;

```

```
158     if ((smq->smq_control & SMQ_VALID_SMQ_MASK) != SMQ_VALID_SMQ)
159         return (SMQ_INVALID);

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

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

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

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

174     smq->smq_head++;
175     /* Increment Head */
176     smq->smq_head = smq->smq_head++;
177     if ((unsigned long)smq->smq_head > ((unsigned long)smq->smq_msgBuffer +
178         (unsigned long)(smq->smq_depth * sizeof (smq_msg_t)))) {
179         smq->smq_head = smq->smq_msgBuffer;
180     }
181     smq->smq_count--;

182     return (0);
183 }
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
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License, Version 1.0 only
6  * (the "License"). You may not use this file except in compliance
7  * with the License.
8  *
9  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
10 * or http://www.opensolaris.org/os/licensing.
11 * See the License for the specific language governing permissions
12 * and limitations under the License.
13 *
14 * When distributing Covered Code, include this CDDL HEADER in each
15 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
16 * If applicable, add the following below this CDDL HEADER, with the
17 * fields enclosed by brackets "[]" replaced with your own identifying
18 * information: Portions Copyright [yyyy] [name of copyright owner]
19 *
20 * CDDL HEADER END
21 */
22 /*
23 * Copyright 2003 Sun Microsystems, Inc. All rights reserved.
24 * Use is subject to license terms.
25 */
27 /*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
28 /*      All Rights Reserved      */

30 /*
31 * University Copyright- Copyright (c) 1982, 1986, 1988
32 * The Regents of the University of California
33 * All Rights Reserved
34 *
35 * University Acknowledgment- Portions of this document are derived from
36 * software developed by the University of California, Berkeley, and its
37 * contributors.
38 */

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

40 #include "tdef.h"
41 #include "tw.h"
42 #include "ext.h"
43 #include <ctype.h>

45 /*
46 * n6.c -- width functions, sizes and fonts
47 */

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

53 extern  int      nchtab;

55 int
56 width(j)
57 tchar j;
58 {
59     int      i, k;

```

```

61     if (j & (ZBIT|MOT)) {
62         if (iszbit(j))
63             return(0);
64         if (isvmot(j))
65             return(0);
66         k = absmot(j);
67         if (isnmot(j))
68             k = -k;
69         return(k);
70     }
71     i = cbits(j);
72     if (i < ' ') {
73         if (i == '\b')
74             return(-widthp);
75         if (i == PRESC)
76             i = eschar;
77         else if (iscontrol(i))
78             return(0);
79     }
80     if (i==ohc)
81         return(0);
82 #ifdef EUC
83 #ifdef NROFF
84     if (multi_locale) {
85         if ((j & MBMASK) || (j & CSMASK)) {
86             switch(j & MBMASK) {
87                 case BYTE_CHR:
88                 case LASTOFMB:
89                     k = t.Char * csi_width[cs(j)];
90                     break;
91                 default:
92                     k = 0;
93                     break;
94             }
95             widthp = k;
96             return(k);
97         }
98     }
99     i &= 0x1fff;
100 #endif /* NROFF */
101 #endif /* EUC */
102     i = trtab[i];
103     if (i < 32)
104         return(0);
105     k = t.width[i] * t.Char;
106     widthp = k;
107     return(k);
108 }
    unchanged_portion_omitted

264 int
265 setwd()
266 {
267     int      base, wid;
268     tchar i;
269     int      delim, emsz, k;
270     int      savhp, savapts, savapts1, savfont, savfont1, savpts, savpts1;

272     base = numtab[ST].val = wid = numtab[CT].val = 0;
274     base = numtab[ST].val = numtab[ST].val = wid = numtab[CT].val = 0;
273     if (ismot(i = getch()))
274         return (0);
275     delim = cbits(i);
276     savhp = numtab[HP].val;

```

```
277     numtab[HP].val = 0;
278     savapts = apts;
279     savapts1 = apts1;
280     savfont = font;
281     savfont1 = font1;
282     savpts = pts;
283     savpts1 = pts1;
284     setwdf++;
285     while (cbits(i = getch()) != delim && !nlflg) {
286         k = width(i);
287         wid += k;
288         numtab[HP].val += k;
289         if (!ismot(i)) {
290             emsz = (INCH * pts + 36) / 72;
291         } else if (isvmot(i)) {
292             k = absmot(i);
293             if (isnmot(i))
294                 k = -k;
295             base -= k;
296             emsz = 0;
297         } else
298             continue;
299         if (base < numtab[SB].val)
300             numtab[SB].val = base;
301         if ((k = base + emsz) > numtab[ST].val)
302             numtab[ST].val = k;
303     }
304     setnl(wid, 0, (tchar) 0);
305     numtab[HP].val = savhp;
306     apts = savapts;
307     apts1 = savapts1;
308     font = savfont;
309     font1 = savfont1;
310     pts = savpts;
311     pts1 = savpts1;
312     mchbits();
313     setwdf = 0;
314
315     return (0);
316 }
```

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
*****
1 /*
2  * CDDL HEADER START
3  *
4  * The contents of this file are subject to the terms of the
5  * Common Development and Distribution License, Version 1.0 only
6  * (the "License"). You may not use this file except in compliance
7  * with the License.
8  *
9  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
10 * or http://www.opensolaris.org/os/licensing.
11 * See the License for the specific language governing permissions
12 * and limitations under the License.
13 *
14 * When distributing Covered Code, include this CDDL HEADER in each
15 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
16 * If applicable, add the following below this CDDL HEADER, with the
17 * fields enclosed by brackets "[]" replaced with your own identifying
18 * information: Portions Copyright [yyyy] [name of copyright owner]
19 *
20 * CDDL HEADER END
21 */
22 /*
23  * Copyright 2003 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25  */
27 /*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
28 /*      All Rights Reserved      */

30 /*
31  * University Copyright- Copyright (c) 1982, 1986, 1988
32  * The Regents of the University of California
33  * All Rights Reserved
34  *
35  * University Acknowledgment- Portions of this document are derived from
36  * software developed by the University of California, Berkeley, and its
37  * contributors.
38  */

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

40 /*
41  * t6.c
42  *
43  * width functions, sizes and fonts
44  */

46 #include "tdef.h"
47 #include "dev.h"
48 #include <ctype.h>
49 #include "ext.h"

51 /* fitab[f][c] is 0 if c is not on font f */
52 /* if it's non-zero, c is in fontab[f] at position
53  * fitab[f][c].
54  */
55 extern struct Font *fontbase[NFONT+1];
56 extern char *codetab[NFONT+1];
57 extern int nctab;

59 int      fontlab[NFONT+1];

```

```

60 short    *pstab;
61 int      cstab[NFONT+1];
62 int      ccstab[NFONT+1];
63 int      bdtab[NFONT+1];
64 int      sbold = 0;

66 int
67 width(j)
68 tchar j;
69 {
70     int      i, k;

72     if (j & (ZBIT|MOT)) {
73         if (iszbit(j))
74             return(0);
75         if (isvmot(j))
76             return(0);
77         k = absmot(j);
78         if (isnmot(j))
79             k = -k;
80         return(k);
81     }
82     i = cbits(j);
83     if (i < ' ') {
84         if (i == '\b')
85             return(-widthp);
86         if (i == PRESC)
87             i = eschar;
88         else if (iscontrol(i))
89             return(0);
90     }
91     if (i==ohc)
92         return(0);
93     i = trtab[i];
94     if (i < 32)
95         return(0);
96     if (sfbits(j) == oldbits) {
97         xfont = pfont;
98         xpts = ppt;
99     } else
100         xbits(j, 0);
101     if (widcache[i-32].fontpts == (xfont<<8) + xpts && !setwdf)
102         k = widcache[i-32].width;
103     else {
104         k = getcw(i-32);
105         if (bd)
106             k += (bd - 1) * HOR;
107         if (cs)
108             k = cs;
109     }
110     widthp = k;
111     return(k);
112 }

unchanged_portion_omitted

493 int
494 setwd()
495 {
496     int      base, wid;
497     tchar i;
498     int      delim, emsz, k;
499     int      savh, savapts, savapts1, savfont, savfont1, savpts, savpts1;

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

```

```
502     if (ismot(i = getch()))
503         return (0);
504     delim = cbits(i);
505     savhp = numtab[HP].val;
506     numtab[HP].val = 0;
507     savapts = apts;
508     savapts1 = apts1;
509     savfont = font;
510     savfont1 = font1;
511     savpts = pts;
512     savpts1 = pts1;
513     setwdf++;
514     while (cbits(i = getch()) != delim && !nlflg) {
515         k = width(i);
516         wid += k;
517         numtab[HP].val += k;
518         if (!ismot(i)) {
519             emsz = POINT * xpts;
520         } else if (isvmot(i)) {
521             k = absmot(i);
522             if (isnmot(i))
523                 k = -k;
524             base -= k;
525             emsz = 0;
526         } else
527             continue;
528         if (base < numtab[SB].val)
529             numtab[SB].val = base;
530         if ((k = base + emsz) > numtab[ST].val)
531             numtab[ST].val = k;
532     }
533     setnl(wid, 0, (tchar) 0);
534     numtab[HP].val = savhp;
535     apts = savapts;
536     apts1 = savapts1;
537     font = savfont;
538     font1 = savfont1;
539     pts = savpts;
540     pts1 = savpts1;
541     mchbits();
542     setwdf = 0;
544     return (0);
545 }
```

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_____

906 static void
907 hxge_ldgv_setup(p_hxge_ldg_t *ldgp, p_hxge_ldv_t *ldvp, uint8_t ldv,
908               uint8_t endlg, int *ngrps)
909 {
910     HXGE_DEBUG_MSG((NULL, INT_CTL, "=> hxge_ldgv_setup"));
911     /* Assign the group number for each device. */
912     (*ldvp)->ldg_assigned = (*ldgp)->ldg;
913     (*ldvp)->ldgp = *ldgp;
914     (*ldvp)->ldv = ldv;

916     HXGE_DEBUG_MSG((NULL, INT_CTL,
917                   "=> hxge_ldgv_setup: ldv %d endlg %d ldg %d, ldvp %p",
918                   ldv, endlg, (*ldgp)->ldg, (*ldgp)->ldvp));

920     (*ldgp)->nldvs++;
921     if ((*ldgp)->ldg == (endlg - 1)) {
922         if ((*ldgp)->ldvp == NULL) {
923             (*ldgp)->ldvp = *ldvp;
924             *ngrps += 1;
925             HXGE_DEBUG_MSG((NULL, INT_CTL,
926                   "=> hxge_ldgv_setup: ngrps %d", *ngrps));
927         }
928         HXGE_DEBUG_MSG((NULL, INT_CTL,
929                   "=> hxge_ldgv_setup: ldvp %p ngrps %d",
930                   *ldvp, *ngrps));
931         ++*ldvp;
932     } else {
933         (*ldgp)->ldvp = *ldvp;
934         *ngrps += 1;
935         HXGE_DEBUG_MSG((NULL, INT_CTL, "=> hxge_ldgv_setup(done): "
936                   "ldv %d endlg %d ldg %d, ldvp %p",
937                   ldv, endlg, (*ldgp)->ldg, (*ldgp)->ldvp));
938         ++*ldvp;
939         ++*ldgp;
940         (*ldvp) = ++*ldvp;
941         (*ldgp) = ++*ldgp;
942         HXGE_DEBUG_MSG((NULL, INT_CTL,
943                   "=> hxge_ldgv_setup: new ngrps %d", *ngrps));
944     }

944     HXGE_DEBUG_MSG((NULL, INT_CTL, "=> hxge_ldgv_setup: "
945                   "ldg %d nldvs %d ldv %d ldvp %p endlg %d ngrps %d",
946                   (*ldgp)->ldg, (*ldgp)->nldvs, ldv, ldvp, endlg, *ngrps));

948     HXGE_DEBUG_MSG((NULL, INT_CTL, "<=> hxge_ldgv_setup"));
949 }
_____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_____

644 /*
645 * hermon_impl_mbox_free()
646 * Context: Can be called from interrupt or base context.
647 */
648 static void
649 hermon_impl_mbox_free(hermon_mboxlist_t *mbl_list, hermon_mbox_t **mb)
650 {
651     uint_t          mbox_indx;
652
653     mutex_enter(&mbl_list->mbl_lock);
654
655     /* Pull the "index" from mailbox entry */
656     mbox_indx = (*mb)->mb_indx;
657
658     /*
659     * If mailbox list is not empty, then insert the entry. Otherwise,
660     * this is the only entry. So update the pointers appropriately.
661     */
662     if (mbl_list->mbl_entries_free++ != 0) {
663         /* Update the current mailbox */
664         (*mb)->mb_next = mbl_list->mbl_head_indx;
665         (*mb)->mb_prev = mbl_list->mbl_tail_indx;
666
667         /* Update head and tail mailboxes */
668         mbl_list->mbl_mbox[mbl_list->mbl_head_indx].mb_prev = mbox_indx;
669         mbl_list->mbl_mbox[mbl_list->mbl_tail_indx].mb_next = mbox_indx;
670
671         /* Update tail index */
672         mbl_list->mbl_tail_indx = mbox_indx;
673
674     } else {
675         /* Update the current mailbox */
676         (*mb)->mb_next = mbox_indx;
677         (*mb)->mb_prev = mbox_indx;
678
679         /* Update head and tail indexes */
680         mbl_list->mbl_tail_indx = mbox_indx;
681         mbl_list->mbl_head_indx = mbox_indx;
682     }
683
684     /*
685     * Because we can have both waiters (SLEEP threads waiting for a
686     * cv_signal to continue processing) and pollers (NOSLEEP threads
687     * polling for a mailbox to become available), we try to share CPU time
688     * between them. We do this by signalling the waiters only every other
689     * call to mbox_free. This gives the pollers a chance to get some CPU
690     * time to do their command. If we signalled every time, the pollers
691     * would have a much harder time getting CPU time.
692     *
693     * If there are waiters and no pollers, then we signal always.
694     *
695     * Otherwise, if there are either no waiters, there may in fact be
696     * pollers, so we do not signal in that case.
697     */
698     if (mbl_list->mbl_pollers > 0 && mbl_list->mbl_waiters > 0) {
699         /* flip the signal value */
700         mbl_list->mbl_signal = (mbl_list->mbl_signal + 1) % 2;
701     }
702     mbl_list->mbl_signal = (++mbl_list->mbl_signal) % 2;

```

```

701     } else if (mbl_list->mbl_waiters > 0) {
702         mbl_list->mbl_signal = 1;
703     } else {
704         mbl_list->mbl_signal = 0;
705     }
706
707     /*
708     * Depending on the conditions in the previous check, we signal only if
709     * we are supposed to.
710     */
711     if (mbl_list->mbl_signal) {
712         mbl_list->mbl_waiters--;
713         cv_signal(&mbl_list->mbl_cv);
714     }
715
716     /* Clear out the mailbox entry pointer */
717     *mb = NULL;
718
719     mutex_exit(&mbl_list->mbl_lock);
720 }
_____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_____

```

```

621 /*
622  * tavor_impl_mbox_free()
623  * Context: Can be called from interrupt or base context.
624  */
625 static void
626 tavor_impl_mbox_free(tavor_mboxlist_t *mblst, tavor_mbox_t **mb)
627 {
628     uint_t          mbox_idx;
629
630     TAVOR_TNF_ENTER(tavor_impl_mbox_free);
631
632     mutex_enter(&mblst->mbl_lock);
633
634     /* Pull the "index" from mailbox entry */
635     mbox_idx = (*mb)->mb_idx;
636
637     /*
638      * If mailbox list is not empty, then insert the entry. Otherwise,
639      * this is the only entry. So update the pointers appropriately.
640      */
641     if (mblst->mbl_entries_free++ != 0) {
642         /* Update the current mailbox */
643         (*mb)->mb_next = mblst->mbl_head_idx;
644         (*mb)->mb_prev = mblst->mbl_tail_idx;
645
646         /* Update head and tail mailboxes */
647         mblst->mbl_mbox[mblst->mbl_head_idx].mb_prev = mbox_idx;
648         mblst->mbl_mbox[mblst->mbl_tail_idx].mb_next = mbox_idx;
649
650         /* Update tail index */
651         mblst->mbl_tail_idx = mbox_idx;
652     } else {
653         /* Update the current mailbox */
654         (*mb)->mb_next = mbox_idx;
655         (*mb)->mb_prev = mbox_idx;
656
657         /* Update head and tail indexes */
658         mblst->mbl_tail_idx = mbox_idx;
659         mblst->mbl_head_idx = mbox_idx;
660     }
661
662     /*
663      * Because we can have both waiters (SLEEP threads waiting for a
664      * cv_signal to continue processing) and pollers (NOSLEEP threads
665      * polling for a mailbox to become available), we try to share CPU time
666      * between them. We do this by signalling the waiters only every other
667      * call to mbox_free. This gives the pollers a chance to get some CPU
668      * time to do their command. If we signalled every time, the pollers
669      * would have a much harder time getting CPU time.
670      *
671      * If there are waiters and no pollers, then we signal always.
672      *
673      * Otherwise, if there are either no waiters, there may in fact be
674      * pollers, so we do not signal in that case.
675      */
676     if (mblst->mbl_pollers > 0 && mblst->mbl_waiters > 0) {
677         /* flip the signal value */

```

```

679         mblst->mbl_signal = (mblst->mbl_signal + 1) % 2;
680         mblst->mbl_signal = (++mblst->mbl_signal) % 2;
681     } else if (mblst->mbl_waiters > 0) {
682         mblst->mbl_signal = 1;
683     } else {
684         mblst->mbl_signal = 0;
685     }
686
687     /*
688      * Depending on the conditions in the previous check, we signal only if
689      * we are supposed to.
690      */
691     if (mblst->mbl_signal) {
692         mblst->mbl_waiters--;
693         cv_signal(&mblst->mbl_cv);
694     }
695
696     /* Clear out the mailbox entry pointer */
697     *mb = NULL;
698
699     mutex_exit(&mblst->mbl_lock);
700
701     TAVOR_TNF_EXIT(tavor_impl_mbox_free);
702 }
_____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_____

3931 /*
3932  * Static functions start here.
3933  */

3935 static void
3936 nxge_ldgv_setup(p_nxge_ldg_t *ldgp, p_nxge_ldv_t *ldvp, uint8_t ldv,
3937                uint8_t endlg, int *ngrps)
3938 {
3939     NXGE_DEBUG_MSG((NULL, INT_CTL, "==> nxge_ldgv_setup"));
3940     /* Assign the group number for each device. */
3941     (*ldvp)->ldg_assigned = (*ldgp)->ldg;
3942     (*ldvp)->ldgp = *ldgp;
3943     (*ldvp)->ldv = ldv;

3945     NXGE_DEBUG_MSG((NULL, INT_CTL, "==> nxge_ldgv_setup: "
3946                    "ldv %d endlg %d ldg %d, ldvp %p",
3947                    ldv, endlg, (*ldgp)->ldg, (*ldgp)->ldvp));

3949     (*ldgp)->nldvs++;
3950     if ((*ldgp)->ldg == (endlg - 1)) {
3951         if ((*ldgp)->ldvp == NULL) {
3952             (*ldgp)->ldvp = *ldvp;
3953             *ngrps += 1;
3954             NXGE_DEBUG_MSG((NULL, INT_CTL,
3955                            "==> nxge_ldgv_setup: ngrps %d", *ngrps));
3956         }
3957         NXGE_DEBUG_MSG((NULL, INT_CTL,
3958                        "==> nxge_ldgv_setup: ldvp %p ngrps %d",
3959                        *ldvp, *ngrps));
3960         ++ldvp;
3961     } else {
3962         (*ldgp)->ldvp = *ldvp;
3963         *ngrps += 1;
3964         NXGE_DEBUG_MSG((NULL, INT_CTL, "==> nxge_ldgv_setup(done): "
3965                        "ldv %d endlg %d ldg %d, ldvp %p",
3966                        ldv, endlg, (*ldgp)->ldg, (*ldgp)->ldvp));
3967         ++ldvp;
3968         ++ldgp;
3969         (*ldvp) = ++ldvp;
3970         (*ldgp) = ++ldgp;
3971         NXGE_DEBUG_MSG((NULL, INT_CTL,
3972                        "==> nxge_ldgv_setup: new ngrps %d", *ngrps));
3973     }

3975     NXGE_DEBUG_MSG((NULL, INT_CTL, "==> nxge_ldgv_setup: "
3976                    "ldv %d ldvp %p endlg %d ngrps %d",
3977                    ldv, ldvp, endlg, *ngrps));

3978     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_____

799 /* fields in diskhd */
800 #define hd_cnt                b_back
801 #define hd_private            b_forw
802 #define hd_flags              b_flags
803 #define hd_sync_next         av_forw
804 #define hd_async_next        av_back

806 #define hd_sync2async         sync_async_ratio

808 #define QNEAR_FORWARD         0x01
809 #define QNEAR_BACKWARD       0x02
810 #define QNEAR_ASYNCONLY      0x04
811 #define QNEAR_ASYNCALSO     0x08

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

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

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

824 /*
825 * qmerge implements a two priority queue, the low priority queue holding ASYNC
826 * write requests, while the rest are queued in the high priority sync queue.
827 * Requests on the async queue would be merged if possible.
828 * By default qmerge2wayscan is 1, indicating an elevator algorithm. When
829 * this variable is set to zero, it has the following side effects.
830 * 1. We assume fairness is the number one issue.
831 * 2. The next request to be picked indicates current head position.
832 *
833 * qmerge_sync2async indicates the ratio of scans of high priority
834 * sync queue to low priority async queue.
835 *
836 * When qmerge variables have the following values it defaults to qsort
837 *
838 * qmergelpri = 1, qmerge2wayscan = 0, qmerge_max_merge = 0
839 *
840 */
841 static int      qmerge_max_merge = 128 * 1024;
842 static intp_t   qmerge_sync2async = 4;
843 static int      qmerge2wayscan = 1;
844 static int      qmergelpri = 0;
845 static int      qmerge_merge = 0;

847 /*
848 *      Local static data
849 */
850 struct que_obj *
851 qmerge_create()
852 {
853     struct que_data *qfp;
854     struct que_obj *queobjp;

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

```

```

858         return (NULL);

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

867     return ((opaque_t)queobjp);
868 }
_____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_____

1773 /*
1774 * sghsc_rb_put()
1775 *     Insert an event info into the event ring buffer.
1776 * Returns DDI_FAILURE if the buffer is full, DDI_SUCCESS otherwise
1777 */
1778 static int
1779 sghsc_rb_put(sghsc_rb_head_t *rb_head, sghsc_event_t *event)
1780 {
1781     if (rb_head->state == SGHSC_RB_FULL)
1782         return (DDI_FAILURE);
1784     rb_head->buf[rb_head->put_idx] = *event;
1786     rb_head->put_idx = (rb_head->put_idx + 1) & (rb_head->size - 1);
1787     rb_head->put_idx = ++rb_head->put_idx & (rb_head->size - 1);
1788     if (rb_head->put_idx == rb_head->get_idx)
1789         rb_head->state = SGHSC_RB_FULL;
1790     else
1791         rb_head->state = SGHSC_RB_FLOAT;
1793     return (DDI_SUCCESS);
1794 }
1795 /*
1796 * sghsc_rb_get()
1797 *     Remove an event info from the event ring buffer.
1798 * Returns DDI_FAILURE if the buffer is empty, DDI_SUCCESS otherwise.
1799 */
1800 static int
1801 sghsc_rb_get(sghsc_rb_head_t *rb_head, sghsc_event_t *event)
1802 {
1804     if (rb_head->state == SGHSC_RB_EMPTY)
1805         return (DDI_FAILURE);
1807     *event = rb_head->buf[rb_head->get_idx];
1809     rb_head->get_idx = (rb_head->get_idx + 1) & (rb_head->size - 1);
1810     rb_head->get_idx = ++rb_head->get_idx & (rb_head->size - 1);
1811     if (rb_head->get_idx == rb_head->put_idx)
1812         rb_head->state = SGHSC_RB_EMPTY;
1813     else
1814         rb_head->state = SGHSC_RB_FLOAT;
1816     return (DDI_SUCCESS);
1817 }
_____unchanged_portion_omitted_____
```