****
119323 Fri May 8 18:04:28 2015
new/usr/src/uts/common/os/lgrp.c
segop_getpolicy already checks for a NULL op
****
_____***unchanged_portion_omitted_***

```
3498 /*
3499  * Get memory allocation policy for this segment
3500  */
3501 lgrp_mem_policy_info_t *
3502 lgrp_mem_policy_get(struct seg *seg, caddr_t vaddr)
3503 {
3504         lgrp_mem_policy_info_t  *policy_info;
3504         extern struct seg_ops   segspt_ops;
3505         extern struct seg_ops   segspt_shmops;

3507         /*
3508          * This is for binary compatibility to protect against third party
3509          * segment drivers which haven't recompiled to allow for
3510          * segop_getpolicy()
3511          */
3512         if (seg->s_ops != &segvn_ops && seg->s_ops != &segspt_ops &&
3513             seg->s_ops != &segspt_shmops)
3514                 return (NULL);

3516         return (segop_getpolicy(seg, vaddr));
3517         policy_info = NULL;
3518         if (seg->s_ops->getpolicy != NULL)
3519                 policy_info = segop_getpolicy(seg, vaddr);

3521         return (policy_info);
3517 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
  113857 Fri May  8 18:04:29 2015
new/usr/src/uts/common/vm/seg_dev.c
segop_getpolicy already checks for a NULL op
**********************************************************
```

```
   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 (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */

  22 /*
  23  * Copyright 2010 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 /*
  41  * VM - segment of a mapped device.
  42  *
  43  * This segment driver is used when mapping character special devices.
  44  */

  46 #include <sys/types.h>
  47 #include <sys/t_lock.h>
  48 #include <sys/sysmacros.h>
  49 #include <sys/vtrace.h>
  50 #include <sys/systm.h>
  51 #include <sys/vmsystm.h>
  52 #include <sys/mman.h>
  53 #include <sys/errno.h>
  54 #include <sys/kmem.h>
  55 #include <sys/cmn_err.h>
  56 #include <sys/vnode.h>
  57 #include <sys/proc.h>
  58 #include <sys/conf.h>
  59 #include <sys/debug.h>
  60 #include <sys/ddidevmap.h>
  61 #include <sys/ddi_implfuncs.h>
```



```
  62 #include <sys/lgrp.h>

  64 #include <vm/page.h>
  65 #include <vm/hat.h>
  66 #include <vm/as.h>
  67 #include <vm/seg.h>
  68 #include <vm/seg_dev.h>
  69 #include <vm/seg_kp.h>
  70 #include <vm/seg_kmem.h>
  71 #include <vm/vpage.h>

  73 #include <sys/sunddi.h>
  74 #include <sys/esunddi.h>
  75 #include <sys/fs/snode.h>


  78 #if DEBUG
  79 int segdev_debug;
  80 #define DEBUGF(level, args) { if (segdev_debug >= (level)) cmn_err args; }
  81 #else
  82 #define DEBUGF(level, args)
  83 #endif

  85 /* Default timeout for devmap context management */
  86 #define CTX_TIMEOUT_VALUE 0

  88 #define HOLD_DHP_LOCK(dhp)  if (dhp->dh_flags & DEVMAP_ALLOW_REMAP) \
  89                         { mutex_enter(&dhp->dh_lock); }

  91 #define RELE_DHP_LOCK(dhp) if (dhp->dh_flags & DEVMAP_ALLOW_REMAP) \
  92                         { mutex_exit(&dhp->dh_lock); }

  94 #define round_down_p2(a, s)     ((a) & ~((s) - 1))
  95 #define round_up_p2(a, s)       (((a) + (s) - 1) & ~((s) - 1))

  97 /*
  98  * VA_PA_ALIGNED checks to see if both VA and PA are on pgsize boundary
  99  * VA_PA_PGSIZE_ALIGNED check to see if VA is aligned with PA w.r.t. pgsize
 100  */
 101 #define VA_PA_ALIGNED(uvaddr, paddr, pgsize)            \
 102         (((uvaddr | paddr) & (pgsize - 1)) == 0)
 103 #define VA_PA_PGSIZE_ALIGNED(uvaddr, paddr, pgsize)     \
 104         (((uvaddr ^ paddr) & (pgsize - 1)) == 0)

 106 #define vpgtob(n)       ((n) * sizeof (struct vpage))   /* For brevity */

 108 #define VTOCVP(vp)      (VTOS(vp)->s_commonvp)  /* we "know" it's an snode */

 110 static struct devmap_ctx *devmapctx_list = NULL;
 111 static struct devmap_softlock *devmap_slist = NULL;

 113 /*
 114  * mutex, vnode and page for the page of zeros we use for the trash mappings.
 115  * One trash page is allocated on the first ddi_umem_setup call that uses it
 116  * XXX Eventually, we may want to combine this with what segnf does when all
 117  * hat layers implement HAT_NOFAULT.
 118  *
 119  * The trash page is used when the backing store for a userland mapping is
 120  * removed but the application semantics do not take kindly to a SIGBUS.
 121  * In that scenario, the applications pages are mapped to some dummy page
 122  * which returns garbage on read and writes go into a common place.
 123  * (Perfect for NO_FAULT semantics)
 124  * The device driver is responsible to communicating to the app with some
 125  * other mechanism that such remapping has happened and the app should take
 126  * corrective action.
 127  * We can also use an anonymous memory page as there is no requirement to
```

```
 128  * keep the page locked, however this complicates the fault code. RFE.
 129  */
 130 static struct vnode trashvp;
 131 static struct page *trashpp;

 133 /* Non-pageable kernel memory is allocated from the umem_np_arena. */
 134 static vmem_t *umem_np_arena;

 136 /* Set the cookie to a value we know will never be a valid umem_cookie */
 137 #define DEVMAP_DEVMEM_COOKIE    ((ddi_umem_cookie_t)0x1)

 139 /*
 140  * Macros to check if type of devmap handle
 141  */
 142 #define cookie_is_devmem(c)     \
 143         ((c) == (struct ddi_umem_cookie *)DEVMAP_DEVMEM_COOKIE)

 145 #define cookie_is_pmem(c)       \
 146         ((c) == (struct ddi_umem_cookie *)DEVMAP_PMEM_COOKIE)

 148 #define cookie_is_kpmem(c)      (!cookie_is_devmem(c) && !cookie_is_pmem(c) &&\
 149         ((c)->type == KMEM_PAGEABLE))

 151 #define dhp_is_devmem(dhp)      \
 152         (cookie_is_devmem((struct ddi_umem_cookie *)((dhp)->dh_cookie)))

 154 #define dhp_is_pmem(dhp)        \
 155         (cookie_is_pmem((struct ddi_umem_cookie *)((dhp)->dh_cookie)))

 157 #define dhp_is_kpmem(dhp)       \
 158         (cookie_is_kpmem((struct ddi_umem_cookie *)((dhp)->dh_cookie)))

 160 /*
 161  * Private seg op routines.
 162  */
 163 static int      segdev_dup(struct seg *, struct seg *);
 164 static int      segdev_unmap(struct seg *, caddr_t, size_t);
 165 static void     segdev_free(struct seg *);
 166 static faultcode_t segdev_fault(struct hat *, struct seg *, caddr_t, size_t,
 167                     enum fault_type, enum seg_rw);
 168 static faultcode_t segdev_faulta(struct seg *, caddr_t);
 169 static int      segdev_setprot(struct seg *, caddr_t, size_t, uint_t);
 170 static int      segdev_checkprot(struct seg *, caddr_t, size_t, uint_t);
 171 static void     segdev_badop(void);
 172 static int      segdev_sync(struct seg *, caddr_t, size_t, int, uint_t);
 173 static size_t   segdev_incore(struct seg *, caddr_t, size_t, char *);
 174 static int      segdev_lockop(struct seg *, caddr_t, size_t, int, int,
 175                     ulong_t *, size_t);
 176 static int      segdev_getprot(struct seg *, caddr_t, size_t, uint_t *);
 177 static u_offset_t       segdev_getoffset(struct seg *, caddr_t);
 178 static int      segdev_gettype(struct seg *, caddr_t);
 179 static int      segdev_getvp(struct seg *, caddr_t, struct vnode **);
 180 static int      segdev_advise(struct seg *, caddr_t, size_t, uint_t);
 181 static void     segdev_dump(struct seg *);
 182 static int      segdev_pagelock(struct seg *, caddr_t, size_t,
 183                     struct page ***, enum lock_type, enum seg_rw);
 184 static int      segdev_setpagesize(struct seg *, caddr_t, size_t, uint_t);
 185 static int      segdev_getmemid(struct seg *, caddr_t, memid_t *);
 186 static lgrp_mem_policy_info_t   *segdev_getpolicy(struct seg *, caddr_t);
 186 static int      segdev_capable(struct seg *, segcapability_t);

 188 /*
 189  * XXX  this struct is used by rootnex_map_fault to identify
 190  *      the segment it has been passed. So if you make it
 191  *      "static" you'll need to fix rootnex_map_fault.
 192  */
```



```
 193 struct seg_ops segdev_ops = {
 194         .dup            = segdev_dup,
 195         .unmap          = segdev_unmap,
 196         .free           = segdev_free,
 197         .fault          = segdev_fault,
 198         .faulta         = segdev_faulta,
 199         .setprot        = segdev_setprot,
 200         .checkprot      = segdev_checkprot,
 201         .kluster        = (int (*)())segdev_badop,
 202         .sync           = segdev_sync,
 203         .incore         = segdev_incore,
 204         .lockop         = segdev_lockop,
 205         .getprot        = segdev_getprot,
 206         .getoffset      = segdev_getoffset,
 207         .gettype        = segdev_gettype,
 208         .getvp          = segdev_getvp,
 209         .advise         = segdev_advise,
 210         .dump           = segdev_dump,
 211         .pagelock       = segdev_pagelock,
 212         .setpagesize    = segdev_setpagesize,
 213         .getmemid       = segdev_getmemid,
 215         .getpolicy      = segdev_getpolicy,
 214         .capable        = segdev_capable,
 215 };
```

_____unchanged_portion_omitted_

```
4016 static int
4017 segdev_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp)
4018 {
4019         struct segdev_data *sdp = (struct segdev_data *)seg->s_data;

4021         /*
4022          * It looks as if it is always mapped shared
4023          */
4024         TRACE_0(TR_FAC_DEVMAP, TR_DEVMAP_GETMEMID,
4025             "segdev_getmemid:start");
4026         memidp->val[0] = (uintptr_t)VTOCVP(sdp->vp);
4027         memidp->val[1] = sdp->offset + (uintptr_t)(addr - seg->s_base);
4028         return (0);
4031 }

4033 /*ARGSUSED*/
4034 static lgrp_mem_policy_info_t *
4035 segdev_getpolicy(struct seg *seg, caddr_t addr)
4036 {
4037         return (NULL);
4029 }
```

_____unchanged_portion_omitted_

```
**********************************************************
   44752 Fri May  8 18:04:29 2015
new/usr/src/uts/common/vm/seg_kmem.c
segop_getpolicy already checks for a NULL op
**********************************************************
______unchanged_portion_omitted_

 752 /*ARGSUSED*/
 753 static lgrp_mem_policy_info_t *
 754 segkmem_getpolicy(struct seg *seg, caddr_t addr)
 755 {
 756         return (NULL);
 757 }

 759 /*ARGSUSED*/
 753 static int
 754 segkmem_capable(struct seg *seg, segcapability_t capability)
 755 {
 756         if (capability == S_CAPABILITY_NOMINFLT)
 757                 return (1);
 758         return (0);
 759 }

 761 static struct seg_ops segkmem_ops = {
 762         .fault          = segkmem_fault,
 763         .setprot        = segkmem_setprot,
 764         .checkprot      = segkmem_checkprot,
 765         .kluster        = segkmem_kluster,
 766         .dump           = segkmem_dump,
 767         .pagelock       = segkmem_pagelock,
 768         .getmemid       = segkmem_getmemid,
 776         .getpolicy      = segkmem_getpolicy,
 769         .capable        = segkmem_capable,
 770 };
______unchanged_portion_omitted_
```

```
**********************************************************
   36054 Fri May  8 18:04:29 2015
new/usr/src/uts/common/vm/seg_kp.c
segop_getpolicy already checks for a NULL op
**********************************************************
```

```
   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 (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright (c) 1991, 2010, Oracle and/or its affiliates. All rights reserved.
  23  */

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

  28 /*
  29  * Portions of this source code were derived from Berkeley 4.3 BSD
  30  * under license from the Regents of the University of California.
  31  */

  33 /*
  34  * segkp is a segment driver that administers the allocation and deallocation
  35  * of pageable variable size chunks of kernel virtual address space. Each
  36  * allocated resource is page-aligned.
  37  *
  38  * The user may specify whether the resource should be initialized to 0,
  39  * include a redzone, or locked in memory.
  40  */

  42 #include <sys/types.h>
  43 #include <sys/t_lock.h>
  44 #include <sys/thread.h>
  45 #include <sys/param.h>
  46 #include <sys/errno.h>
  47 #include <sys/sysmacros.h>
  48 #include <sys/systm.h>
  49 #include <sys/buf.h>
  50 #include <sys/mman.h>
  51 #include <sys/vnode.h>
  52 #include <sys/cmn_err.h>
  53 #include <sys/swap.h>
  54 #include <sys/tuneable.h>
  55 #include <sys/kmem.h>
  56 #include <sys/vmem.h>
  57 #include <sys/cred.h>
  58 #include <sys/dumphdr.h>
  59 #include <sys/debug.h>
  60 #include <sys/vtrace.h>
  61 #include <sys/stack.h>
```



```
  62 #include <sys/atomic.h>
  63 #include <sys/archsystm.h>
  64 #include <sys/lgrp.h>

  66 #include <vm/as.h>
  67 #include <vm/seg.h>
  68 #include <vm/seg_kp.h>
  69 #include <vm/seg_kmem.h>
  70 #include <vm/anon.h>
  71 #include <vm/page.h>
  72 #include <vm/hat.h>
  73 #include <sys/bitmap.h>

  75 /*
  76  * Private seg op routines
  77  */
  78 static void     segkp_dump(struct seg *seg);
  79 static int      segkp_checkprot(struct seg *seg, caddr_t addr, size_t len,
  80                         uint_t prot);
  81 static int      segkp_kluster(struct seg *seg, caddr_t addr, ssize_t delta);
  82 static int      segkp_pagelock(struct seg *seg, caddr_t addr, size_t len,
  83                         struct page ***page, enum lock_type type,
  84                         enum seg_rw rw);
  85 static void     segkp_insert(struct seg *seg, struct segkp_data *kpd);
  86 static void     segkp_delete(struct seg *seg, struct segkp_data *kpd);
  87 static caddr_t  segkp_get_internal(struct seg *seg, size_t len, uint_t flags,
  88                         struct segkp_data **tkpd, struct anon_map *amp);
  89 static void     segkp_release_internal(struct seg *seg,
  90                         struct segkp_data *kpd, size_t len);
  91 static int      segkp_unlock(struct hat *hat, struct seg *seg, caddr_t vaddr,
  92                         size_t len, struct segkp_data *kpd, uint_t flags);
  93 static int      segkp_load(struct hat *hat, struct seg *seg, caddr_t vaddr,
  94                         size_t len, struct segkp_data *kpd, uint_t flags);
  95 static struct   segkp_data *segkp_find(struct seg *seg, caddr_t vaddr);
  96 static int      segkp_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp);
  97 static lgrp_mem_policy_info_t   *segkp_getpolicy(struct seg *seg,
  98     caddr_t addr);
  97 static int      segkp_capable(struct seg *seg, segcapability_t capability);

  99 /*
 100  * Lock used to protect the hash table(s) and caches.
 101  */
 102 static kmutex_t segkp_lock;

 104 /*
 105  * The segkp caches
 106  */
 107 static struct segkp_cache segkp_cache[SEGKP_MAX_CACHE];

 109 /*
 110  * When there are fewer than red_minavail bytes left on the stack,
 111  * segkp_map_red() will map in the redzone (if called).  5000 seems
 112  * to work reasonably well...
 113  */
 114 long            red_minavail = 5000;

 116 /*
 117  * will be set to 1 for 32 bit x86 systems only, in startup.c
 118  */
 119 int     segkp_fromheap = 0;
 120 ulong_t *segkp_bitmap;

 122 /*
 123  * If segkp_map_red() is called with the redzone already mapped and
 124  * with less than RED_DEEP_THRESHOLD bytes available on the stack,
 125  * then the stack situation has become quite serious;  if much more stack
```

```
 126  * is consumed, we have the potential of scrogging the next thread/LWP
 127  * structure.  To help debug the "can't happen" panics which may
 128  * result from this condition, we record hrestime and the calling thread
 129  * in red_deep_hires and red_deep_thread respectively.
 130  */
 131 #define RED_DEEP_THRESHOLD      2000

 133 hrtime_t        red_deep_hires;
 134 kthread_t       *red_deep_thread;

 136 uint32_t        red_nmapped;
 137 uint32_t        red_closest = UINT_MAX;
 138 uint32_t        red_ndoubles;

 140 pgcnt_t anon_segkp_pages_locked;        /* See vm/anon.h */
 141 pgcnt_t anon_segkp_pages_resv;          /* anon reserved by seg_kp */

 143 static struct   seg_ops segkp_ops = {
 144         .fault          = segkp_fault,
 145         .checkprot      = segkp_checkprot,
 146         .kluster        = segkp_kluster,
 147         .dump           = segkp_dump,
 148         .pagelock       = segkp_pagelock,
 149         .getmemid       = segkp_getmemid,
 152         .getpolicy      = segkp_getpolicy,
 150         .capable        = segkp_capable,
 151 };
_____unchanged_portion_omitted_

1366 /*ARGSUSED*/
1367 static int
1368 segkp_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp)
1369 {
1370         return (ENODEV);
1374 }

1376 /*ARGSUSED*/
1377 static lgrp_mem_policy_info_t   *
1378 segkp_getpolicy(struct seg *seg, caddr_t addr)
1379 {
1380         return (NULL);
1371 }
_____unchanged_portion_omitted_
```

```
**********************************************************
   57467 Fri May  8 18:04:30 2015
new/usr/src/uts/common/vm/seg_map.c
segop_getpolicy already checks for a NULL op
**********************************************************
```

```
   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 (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
  23  * Use is subject to license terms.
  24  */

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

  29 /*
  30  * Portions of this source code were derived from Berkeley 4.3 BSD
  31  * under license from the Regents of the University of California.
  32  */

  34 /*
  35  * VM - generic vnode mapping segment.
  36  *
  37  * The segmap driver is used only by the kernel to get faster (than seg_vn)
  38  * mappings [lower routine overhead; more persistent cache] to random
  39  * vnode/offsets.  Note than the kernel may (and does) use seg_vn as well.
  40  */

  42 #include <sys/types.h>
  43 #include <sys/t_lock.h>
  44 #include <sys/param.h>
  45 #include <sys/sysmacros.h>
  46 #include <sys/buf.h>
  47 #include <sys/systm.h>
  48 #include <sys/vnode.h>
  49 #include <sys/mman.h>
  50 #include <sys/errno.h>
  51 #include <sys/cred.h>
  52 #include <sys/kmem.h>
  53 #include <sys/vtrace.h>
  54 #include <sys/cmn_err.h>
  55 #include <sys/debug.h>
  56 #include <sys/thread.h>
  57 #include <sys/dumphdr.h>
  58 #include <sys/bitmap.h>
  59 #include <sys/lgrp.h>

  61 #include <vm/seg_kmem.h>
```



```
  62 #include <vm/hat.h>
  63 #include <vm/as.h>
  64 #include <vm/seg.h>
  65 #include <vm/seg_kpm.h>
  66 #include <vm/seg_map.h>
  67 #include <vm/page.h>
  68 #include <vm/pvn.h>
  69 #include <vm/rm.h>

  71 /*
  72  * Private seg op routines.
  73  */
  74 static void     segmap_free(struct seg *seg);
  75 faultcode_t segmap_fault(struct hat *hat, struct seg *seg, caddr_t addr,
  76                         size_t len, enum fault_type type, enum seg_rw rw);
  77 static faultcode_t segmap_faulta(struct seg *seg, caddr_t addr);
  78 static int      segmap_checkprot(struct seg *seg, caddr_t addr, size_t len,
  79                         uint_t prot);
  80 static int      segmap_kluster(struct seg *seg, caddr_t addr, ssize_t);
  81 static int      segmap_getprot(struct seg *seg, caddr_t addr, size_t len,
  82                         uint_t *protv);
  83 static u_offset_t       segmap_getoffset(struct seg *seg, caddr_t addr);
  84 static int      segmap_gettype(struct seg *seg, caddr_t addr);
  85 static int      segmap_getvp(struct seg *seg, caddr_t addr, struct vnode **vpp);
  86 static void     segmap_dump(struct seg *seg);
  87 static int      segmap_pagelock(struct seg *seg, caddr_t addr, size_t len,
  88                         struct page ***ppp, enum lock_type type,
  89                         enum seg_rw rw);
  90 static int      segmap_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp);
  91 static lgrp_mem_policy_info_t   *segmap_getpolicy(struct seg *seg,
  92     caddr_t addr);
  91 static int      segmap_capable(struct seg *seg, segcapability_t capability);

  93 /* segkpm support */
  94 static caddr_t  segmap_pagecreate_kpm(struct seg *, vnode_t *, u_offset_t,
  95                         struct smap *, enum seg_rw);
  96 struct smap     *get_smap_kpm(caddr_t, page_t **);

  98 static struct seg_ops segmap_ops = {
  99         .free           = segmap_free,
 100         .fault          = segmap_fault,
 101         .faulta         = segmap_faulta,
 102         .checkprot      = segmap_checkprot,
 103         .kluster        = segmap_kluster,
 104         .getprot        = segmap_getprot,
 105         .getoffset      = segmap_getoffset,
 106         .gettype        = segmap_gettype,
 107         .getvp          = segmap_getvp,
 108         .dump           = segmap_dump,
 109         .pagelock       = segmap_pagelock,
 110         .getmemid       = segmap_getmemid,
 113         .getpolicy      = segmap_getpolicy,
 111         .capable        = segmap_capable,
 112 };
_____unchanged_portion_omitted_

2161 static int
2162 segmap_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp)
2163 {
2164         struct segmap_data *smd = (struct segmap_data *)seg->s_data;

2166         memidp->val[0] = (uintptr_t)smd->smd_sm->sm_vp;
2167         memidp->val[1] = smd->smd_sm->sm_off + (uintptr_t)(addr - seg->s_base);
2168         return (0);
2172 }
```

```
2174 /*ARGSUSED*/
2175 static lgrp_mem_policy_info_t *
2176 segmap_getpolicy(struct seg *seg, caddr_t addr)
2177 {
2178         return (NULL);
2169 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   16820 Fri May  8 18:04:30 2015
new/usr/src/uts/i86xpv/vm/seg_mf.c
segop_getpolicy already checks for a NULL op
**********************************************************
______unchanged_portion_omitted_

 502 /*ARGSUSED*/
 503 static lgrp_mem_policy_info_t *
 504 segmf_getpolicy(struct seg *seg, caddr_t addr)
 505 {
 506         return (NULL);
 507 }

 509 /*ARGSUSED*/
 503 static int
 504 segmf_capable(struct seg *seg, segcapability_t capability)
 505 {
 506         return (0);
 507 }
______unchanged_portion_omitted_

 753 static struct seg_ops segmf_ops = {
 754         .dup            = segmf_dup,
 755         .unmap          = segmf_unmap,
 756         .free           = segmf_free,
 757         .fault          = segmf_fault,
 758         .faulta         = segmf_faulta,
 759         .setprot        = segmf_setprot,
 760         .checkprot      = segmf_checkprot,
 761         .kluster        = segmf_kluster,
 762         .sync           = segmf_sync,
 763         .incore         = segmf_incore,
 764         .lockop         = segmf_lockop,
 765         .getprot        = segmf_getprot,
 766         .getoffset      = segmf_getoffset,
 767         .gettype        = segmf_gettype,
 768         .getvp          = segmf_getvp,
 769         .advise         = segmf_advise,
 770         .dump           = segmf_dump,
 771         .pagelock       = segmf_pagelock,
 772         .setpagesize    = segmf_setpagesize,
 773         .getmemid       = segmf_getmemid,
 781         .getpolicy      = segmf_getpolicy,
 774         .capable        = segmf_capable,
 775 };
______unchanged_portion_omitted_
```

```
**********************************************************
   12103 Fri May  8 18:04:30 2015
new/usr/src/uts/sparc/v9/vm/seg_nf.c
segop_getpolicy already checks for a NULL op
**********************************************************
```

```
   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 (the "License").
   6  * You may not use this file except in compliance with the License.
   7  *
   8  * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
   9  * or http://www.opensolaris.org/os/licensing.
  10  * See the License for the specific language governing permissions
  11  * and limitations under the License.
  12  *
  13  * When distributing Covered Code, include this CDDL HEADER in each
  14  * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
  15  * If applicable, add the following below this CDDL HEADER, with the
  16  * fields enclosed by brackets "[]" replaced with your own identifying
  17  * information: Portions Copyright [yyyy] [name of copyright owner]
  18  *
  19  * CDDL HEADER END
  20  */
  21 /*
  22  * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
  23  * Use is subject to license terms.
  24  */

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

  29 /*
  30  * Portions of this source code were derived from Berkeley 4.3 BSD
  31  * under license from the Regents of the University of California.
  32  */

  34 /*
  35  * VM - segment for non-faulting loads.
  36  */

  38 #include <sys/types.h>
  39 #include <sys/t_lock.h>
  40 #include <sys/param.h>
  41 #include <sys/mman.h>
  42 #include <sys/errno.h>
  43 #include <sys/kmem.h>
  44 #include <sys/cmn_err.h>
  45 #include <sys/vnode.h>
  46 #include <sys/proc.h>
  47 #include <sys/conf.h>
  48 #include <sys/debug.h>
  49 #include <sys/archsystm.h>
  50 #include <sys/lgrp.h>

  52 #include <vm/page.h>
  53 #include <vm/hat.h>
  54 #include <vm/as.h>
  55 #include <vm/seg.h>
  56 #include <vm/vpage.h>

  58 /*
  59  * Private seg op routines.
  60  */
  61 static int      segnf_dup(struct seg *seg, struct seg *newseg);
```



```
  62 static int      segnf_unmap(struct seg *seg, caddr_t addr, size_t len);
  63 static void     segnf_free(struct seg *seg);
  64 static faultcode_t segnf_nomap(void);
  65 static int      segnf_setprot(struct seg *seg, caddr_t addr,
  66                     size_t len, uint_t prot);
  67 static int      segnf_checkprot(struct seg *seg, caddr_t addr,
  68                     size_t len, uint_t prot);
  69 static int      segnf_nop(void);
  70 static int      segnf_getprot(struct seg *seg, caddr_t addr,
  71                     size_t len, uint_t *protv);
  72 static u_offset_t segnf_getoffset(struct seg *seg, caddr_t addr);
  73 static int      segnf_gettype(struct seg *seg, caddr_t addr);
  74 static int      segnf_getvp(struct seg *seg, caddr_t addr, struct vnode **vpp);
  75 static void     segnf_dump(struct seg *seg);
  76 static int      segnf_pagelock(struct seg *seg, caddr_t addr, size_t len,
  77                     struct page ***ppp, enum lock_type type, enum seg_rw rw);
  78 static int      segnf_setpagesize(struct seg *seg, caddr_t addr, size_t len,
  79                     uint_t szc);
  80 static int      segnf_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp);
  81 static lgrp_mem_policy_info_t   *segnf_getpolicy(struct seg *seg,
  82     caddr_t addr);


  83 struct seg_ops segnf_ops = {
  84         .dup            = segnf_dup,
  85         .unmap          = segnf_unmap,
  86         .free           = segnf_free,
  87         .fault          = (faultcode_t (*)(struct hat *, struct seg *, caddr_t,
  88             size_t, enum fault_type, enum seg_rw))segnf_nomap,
  89         .faulta         = (faultcode_t (*)(struct seg *, caddr_t)) segnf_nomap,
  90         .setprot        = segnf_setprot,
  91         .checkprot      = segnf_checkprot,
  92         .sync           = (int (*)(struct seg *, caddr_t, size_t, int, uint_t))
  93                 segnf_nop,
  94         .incore         = (size_t (*)(struct seg *, caddr_t, size_t, char *))
  95                 segnf_nop,
  96         .lockop         = (int (*)(struct seg *, caddr_t, size_t, int, int,
  97             ulong_t *, size_t))segnf_nop,
  98         .getprot        = segnf_getprot,
  99         .getoffset      = segnf_getoffset,
 100         .gettype        = segnf_gettype,
 101         .getvp          = segnf_getvp,
 102         .advise         = (int (*)(struct seg *, caddr_t, size_t, uint_t))
 103                 segnf_nop,
 104         .dump           = segnf_dump,
 105         .pagelock       = segnf_pagelock,
 106         .setpagesize    = segnf_setpagesize,
 107         .getmemid       = segnf_getmemid,
 110         .getpolicy      = segnf_getpolicy,
 108 };
______unchanged_portion_omitted_

 469 /*ARGSUSED*/
 470 static int
 471 segnf_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp)
 472 {
 473         return (ENODEV);
 477 }

 479 /*ARGSUSED*/
 480 static lgrp_mem_policy_info_t *
 481 segnf_getpolicy(struct seg *seg, caddr_t addr)
 482 {
 483         return (NULL);
 474 }
______unchanged_portion_omitted_
```