```
**********************************************************
  119323 Tue Nov 24 09:34:55 2015
new/usr/src/uts/common/os/lgrp.c
6147 segop_getpolicy already checks for a NULL op
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
______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 Tue Nov 24 09:34:56 2015
new/usr/src/uts/common/vm/seg_dev.c
6147 segop_getpolicy already checks for a NULL op
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */

/*
 * Copyright 2010 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.
 */

/*
 * VM - segment of a mapped device.
 *
 * This segment driver is used when mapping character special devices.
 */

#include <sys/types.h>
#include <sys/t_lock.h>
#include <sys/sysmacros.h>
#include <sys/vtrace.h>
#include <sys/systm.h>
#include <sys/vmsystm.h>
#include <sys/mman.h>
#include <sys/errno.h>
#include <sys/kmem.h>
#include <sys/cmn_err.h>
#include <sys/vnode.h>
#include <sys/proc.h>
#include <sys/conf.h>
#include <sys/debug.h>
#include <sys/ddidevmap.h>
```



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

#include <vm/page.h>
#include <vm/hat.h>
#include <vm/as.h>
#include <vm/seg.h>
#include <vm/seg_dev.h>
#include <vm/seg_kp.h>
#include <vm/seg_kmem.h>
#include <vm/vpage.h>

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


#if DEBUG
int segdev_debug;
#define DEBUGF(level, args) { if (segdev_debug >= (level)) cmn_err args; }
#else
#define DEBUGF(level, args)
#endif

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

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

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

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

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

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

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

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

/*
 * mutex, vnode and page for the page of zeros we use for the trash mappings.
 * One trash page is allocated on the first ddi_umem_setup call that uses it
 * XXX Eventually, we may want to combine this with what segnf does when all
 * hat layers implement HAT_NOFAULT.
 *
 * The trash page is used when the backing store for a userland mapping is
 * removed but the application semantics do not take kindly to a SIGBUS.
 * In that scenario, the applications pages are mapped to some dummy page
 * which returns garbage on read and writes go into a common place.
 * (Perfect for NO_FAULT semantics)
 * The device driver is responsible to communicating to the app with some
 * other mechanism that such remapping has happened and the app should take
 * 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_
```

```
**********************************************************
   45283 Tue Nov 24 09:34:56 2015
new/usr/src/uts/common/vm/seg_kmem.c
6147 segop_getpolicy already checks for a NULL op
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
______unchanged_portion_omitted_

 760 /*ARGSUSED*/
 761 static lgrp_mem_policy_info_t *
 762 segkmem_getpolicy(struct seg *seg, caddr_t addr)
 763 {
 764         return (NULL);
 765 }

 767 /*ARGSUSED*/
 761 static int
 762 segkmem_capable(struct seg *seg, segcapability_t capability)
 763 {
 764         if (capability == S_CAPABILITY_NOMINFLT)
 765                 return (1);
 766         return (0);
 767 }

 769 static struct seg_ops segkmem_ops = {
 770         .dup            = SEGKMEM_BADOP(int),
 771         .unmap          = SEGKMEM_BADOP(int),
 772         .free           = SEGKMEM_BADOP(void),
 773         .fault          = segkmem_fault,
 774         .faulta         = SEGKMEM_BADOP(faultcode_t),
 775         .setprot        = segkmem_setprot,
 776         .checkprot      = segkmem_checkprot,
 777         .kluster        = segkmem_kluster,
 778         .swapout        = SEGKMEM_BADOP(size_t),
 779         .sync           = SEGKMEM_BADOP(int),
 780         .incore         = SEGKMEM_BADOP(size_t),
 781         .lockop         = SEGKMEM_BADOP(int),
 782         .getprot        = SEGKMEM_BADOP(int),
 783         .getoffset      = SEGKMEM_BADOP(u_offset_t),
 784         .gettype        = SEGKMEM_BADOP(int),
 785         .getvp          = SEGKMEM_BADOP(int),
 786         .advise         = SEGKMEM_BADOP(int),
 787         .dump           = segkmem_dump,
 788         .pagelock       = segkmem_pagelock,
 789         .setpagesize    = SEGKMEM_BADOP(int),
 790         .getmemid       = segkmem_getmemid,
 798         .getpolicy      = segkmem_getpolicy,
 791         .capable        = segkmem_capable,
 792 };
______unchanged_portion_omitted_
```

```
**********************************************************
   36908 Tue Nov 24 09:34:56 2015
new/usr/src/uts/common/vm/seg_kp.c
6147 segop_getpolicy already checks for a NULL op
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright (c) 1991, 2010, Oracle and/or its affiliates. All rights reserved.
 */

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

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

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

#include <sys/types.h>
#include <sys/t_lock.h>
#include <sys/thread.h>
#include <sys/param.h>
#include <sys/errno.h>
#include <sys/sysmacros.h>
#include <sys/systm.h>
#include <sys/buf.h>
#include <sys/mman.h>
#include <sys/vnode.h>
#include <sys/cmn_err.h>
#include <sys/swap.h>
#include <sys/tuneable.h>
#include <sys/kmem.h>
#include <sys/vmem.h>
#include <sys/cred.h>
#include <sys/dumphdr.h>
#include <sys/debug.h>
#include <sys/vtrace.h>
```



```
#include <sys/stack.h>
#include <sys/atomic.h>
#include <sys/archsystm.h>
#include <sys/lgrp.h>

#include <vm/as.h>
#include <vm/seg.h>
#include <vm/seg_kp.h>
#include <vm/seg_kmem.h>
#include <vm/anon.h>
#include <vm/page.h>
#include <vm/hat.h>
#include <sys/bitmap.h>

/*
 * Private seg op routines
 */
static void     segkp_badop(void);
static void     segkp_dump(struct seg *seg);
static int      segkp_checkprot(struct seg *seg, caddr_t addr, size_t len,
                        uint_t prot);
static int      segkp_kluster(struct seg *seg, caddr_t addr, ssize_t delta);
static int      segkp_pagelock(struct seg *seg, caddr_t addr, size_t len,
                        struct page ***page, enum lock_type type,
                        enum seg_rw rw);
static void     segkp_insert(struct seg *seg, struct segkp_data *kpd);
static void     segkp_delete(struct seg *seg, struct segkp_data *kpd);
static caddr_t  segkp_get_internal(struct seg *seg, size_t len, uint_t flags,
                        struct segkp_data **tkpd, struct anon_map *amp);
static void     segkp_release_internal(struct seg *seg,
                        struct segkp_data *kpd, size_t len);
static int      segkp_unlock(struct hat *hat, struct seg *seg, caddr_t vaddr,
                        size_t len, struct segkp_data *kpd, uint_t flags);
static int      segkp_load(struct hat *hat, struct seg *seg, caddr_t vaddr,
                        size_t len, struct segkp_data *kpd, uint_t flags);
static struct   segkp_data *segkp_find(struct seg *seg, caddr_t vaddr);
static int      segkp_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp);
static lgrp_mem_policy_info_t   *segkp_getpolicy(struct seg *seg,
    caddr_t addr);
static int      segkp_capable(struct seg *seg, segcapability_t capability);

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

/*
 * The segkp caches
 */
static struct segkp_cache segkp_cache[SEGKP_MAX_CACHE];

#define SEGKP_BADOP(t)  (t(*)())segkp_badop

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

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

```
125 /*
126  * If segkp_map_red() is called with the redzone already mapped and
127  * with less than RED_DEEP_THRESHOLD bytes available on the stack,
128  * then the stack situation has become quite serious;  if much more stack
129  * is consumed, we have the potential of scrogging the next thread/LWP
130  * structure.  To help debug the "can't happen" panics which may
131  * result from this condition, we record hrestime and the calling thread
132  * in red_deep_hires and red_deep_thread respectively.
133  */
134 #define RED_DEEP_THRESHOLD      2000

136 hrtime_t        red_deep_hires;
137 kthread_t       *red_deep_thread;

139 uint32_t        red_nmapped;
140 uint32_t        red_closest = UINT_MAX;
141 uint32_t        red_ndoubles;

143 pgcnt_t anon_segkp_pages_locked;        /* See vm/anon.h */
144 pgcnt_t anon_segkp_pages_resv;          /* anon reserved by seg_kp */

146 static struct   seg_ops segkp_ops = {
147         .dup            = SEGKP_BADOP(int),
148         .unmap          = SEGKP_BADOP(int),
149         .free           = SEGKP_BADOP(void),
150         .fault          = segkp_fault,
151         .faulta         = SEGKP_BADOP(faultcode_t),
152         .setprot        = SEGKP_BADOP(int),
153         .checkprot      = segkp_checkprot,
154         .kluster        = segkp_kluster,
155         .swapout        = SEGKP_BADOP(size_t),
156         .sync           = SEGKP_BADOP(int),
157         .incore         = SEGKP_BADOP(size_t),
158         .lockop         = SEGKP_BADOP(int),
159         .getprot        = SEGKP_BADOP(int),
160         .getoffset      = SEGKP_BADOP(u_offset_t),
161         .gettype        = SEGKP_BADOP(int),
162         .getvp          = SEGKP_BADOP(int),
163         .advise         = SEGKP_BADOP(int),
164         .dump           = segkp_dump,
165         .pagelock       = segkp_pagelock,
166         .setpagesize    = SEGKP_BADOP(int),
167         .getmemid       = segkp_getmemid,
170         .getpolicy      = segkp_getpolicy,
168         .capable        = segkp_capable,
169 };
______unchanged_portion_omitted_

1398 /*ARGSUSED*/
1399 static int
1400 segkp_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp)
1401 {
1402         return (ENODEV);
1406 }

1408 /*ARGSUSED*/
1409 static lgrp_mem_policy_info_t   *
1410 segkp_getpolicy(struct seg *seg, caddr_t addr)
1411 {
1412         return (NULL);
1403 }
______unchanged_portion_omitted_
```

```
**********************************************************
   57900 Tue Nov 24 09:34:56 2015
new/usr/src/uts/common/vm/seg_map.c
6147 segop_getpolicy already checks for a NULL op
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

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

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

#include <sys/types.h>
#include <sys/t_lock.h>
#include <sys/param.h>
#include <sys/sysmacros.h>
#include <sys/buf.h>
#include <sys/systm.h>
#include <sys/vnode.h>
#include <sys/mman.h>
#include <sys/errno.h>
#include <sys/cred.h>
#include <sys/kmem.h>
#include <sys/vtrace.h>
#include <sys/cmn_err.h>
#include <sys/debug.h>
#include <sys/thread.h>
#include <sys/dumphdr.h>
#include <sys/bitmap.h>
#include <sys/lgrp.h>
```



```
#include <vm/seg_kmem.h>
#include <vm/hat.h>
#include <vm/as.h>
#include <vm/seg.h>
#include <vm/seg_kpm.h>
#include <vm/seg_map.h>
#include <vm/page.h>
#include <vm/pvn.h>
#include <vm/rm.h>

/*
 * Private seg op routines.
 */
static void     segmap_free(struct seg *seg);
faultcode_t segmap_fault(struct hat *hat, struct seg *seg, caddr_t addr,
                        size_t len, enum fault_type type, enum seg_rw rw);
static faultcode_t segmap_faulta(struct seg *seg, caddr_t addr);
static int      segmap_checkprot(struct seg *seg, caddr_t addr, size_t len,
                        uint_t prot);
static int      segmap_kluster(struct seg *seg, caddr_t addr, ssize_t);
static int      segmap_getprot(struct seg *seg, caddr_t addr, size_t len,
                        uint_t *protv);
static u_offset_t       segmap_getoffset(struct seg *seg, caddr_t addr);
static int      segmap_gettype(struct seg *seg, caddr_t addr);
static int      segmap_getvp(struct seg *seg, caddr_t addr, struct vnode **vpp);
static void     segmap_dump(struct seg *seg);
static int      segmap_pagelock(struct seg *seg, caddr_t addr, size_t len,
                        struct page ***ppp, enum lock_type type,
                        enum seg_rw rw);
static void     segmap_badop(void);
static int      segmap_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp);
static lgrp_mem_policy_info_t   *segmap_getpolicy(struct seg *seg,
    caddr_t addr);
static int      segmap_capable(struct seg *seg, segcapability_t capability);

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

#define SEGMAP_BADOP(t) (t(*)())segmap_badop

static struct seg_ops segmap_ops = {
        .dup            = SEGMAP_BADOP(int),
        .unmap          = SEGMAP_BADOP(int),
        .free           = segmap_free,
        .fault          = segmap_fault,
        .faulta         = segmap_faulta,
        .setprot        = SEGMAP_BADOP(int),
        .checkprot      = segmap_checkprot,
        .kluster        = segmap_kluster,
        .swapout        = SEGMAP_BADOP(size_t),
        .sync           = SEGMAP_BADOP(int),
        .incore         = SEGMAP_BADOP(size_t),
        .lockop         = SEGMAP_BADOP(int),
        .getprot        = segmap_getprot,
        .getoffset      = segmap_getoffset,
        .gettype        = segmap_gettype,
        .getvp          = segmap_getvp,
        .advise         = SEGMAP_BADOP(int),
        .dump           = segmap_dump,
        .pagelock       = segmap_pagelock,
        .setpagesize    = SEGMAP_BADOP(int),
        .getmemid       = segmap_getmemid,
        .getpolicy      = segmap_getpolicy,
        .capable        = segmap_capable,
```

```
 124 };
______unchanged_portion_omitted_

2180 static int
2181 segmap_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp)
2182 {
2183         struct segmap_data *smd = (struct segmap_data *)seg->s_data;

2185         memidp->val[0] = (uintptr_t)smd->smd_sm->sm_vp;
2186         memidp->val[1] = smd->smd_sm->sm_off + (uintptr_t)(addr - seg->s_base);
2187         return (0);
2191 }

2193 /*ARGSUSED*/
2194 static lgrp_mem_policy_info_t *
2195 segmap_getpolicy(struct seg *seg, caddr_t addr)
2196 {
2197         return (NULL);
2188 }
______unchanged_portion_omitted_
```

```
**********************************************************
   16820 Tue Nov 24 09:34:56 2015
new/usr/src/uts/i86xpv/vm/seg_mf.c
6147 segop_getpolicy already checks for a NULL op
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
______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_
```

```
**********************************************************
   12244 Tue Nov 24 09:34:56 2015
new/usr/src/uts/sparc/v9/vm/seg_nf.c
6147 segop_getpolicy already checks for a NULL op
Reviewed by: Garrett D'Amore <garrett@damore.org>
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License (the "License").
 * You may not use this file except in compliance with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

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

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

/*
 * VM - segment for non-faulting loads.
 */

#include <sys/types.h>
#include <sys/t_lock.h>
#include <sys/param.h>
#include <sys/mman.h>
#include <sys/errno.h>
#include <sys/kmem.h>
#include <sys/cmn_err.h>
#include <sys/vnode.h>
#include <sys/proc.h>
#include <sys/conf.h>
#include <sys/debug.h>
#include <sys/archsystm.h>
#include <sys/lgrp.h>

#include <vm/page.h>
#include <vm/hat.h>
#include <vm/as.h>
#include <vm/seg.h>
#include <vm/vpage.h>

/*
 * Private seg op routines.
 */
```



```
static int      segnf_dup(struct seg *seg, struct seg *newseg);
static int      segnf_unmap(struct seg *seg, caddr_t addr, size_t len);
static void     segnf_free(struct seg *seg);
static faultcode_t segnf_nomap(void);
static int      segnf_setprot(struct seg *seg, caddr_t addr,
                    size_t len, uint_t prot);
static int      segnf_checkprot(struct seg *seg, caddr_t addr,
                    size_t len, uint_t prot);
static void     segnf_badop(void);
static int      segnf_nop(void);
static int      segnf_getprot(struct seg *seg, caddr_t addr,
                    size_t len, uint_t *protv);
static u_offset_t segnf_getoffset(struct seg *seg, caddr_t addr);
static int      segnf_gettype(struct seg *seg, caddr_t addr);
static int      segnf_getvp(struct seg *seg, caddr_t addr, struct vnode **vpp);
static void     segnf_dump(struct seg *seg);
static int      segnf_pagelock(struct seg *seg, caddr_t addr, size_t len,
                    struct page ***ppp, enum lock_type type, enum seg_rw rw);
static int      segnf_setpagesize(struct seg *seg, caddr_t addr, size_t len,
                    uint_t szc);
static int      segnf_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp);
static lgrp_mem_policy_info_t   *segnf_getpolicy(struct seg *seg,
    caddr_t addr);


struct seg_ops segnf_ops = {
        .dup            = segnf_dup,
        .unmap          = segnf_unmap,
        .free           = segnf_free,
        .fault          = (faultcode_t (*)(struct hat *, struct seg *, caddr_t,
            size_t, enum fault_type, enum seg_rw))segnf_nomap,
        .faulta         = (faultcode_t (*)(struct seg *, caddr_t)) segnf_nomap,
        .setprot        = segnf_setprot,
        .checkprot      = segnf_checkprot,
        .kluster        = (int (*)())segnf_badop,
        .sync           = (int (*)(struct seg *, caddr_t, size_t, int, uint_t))
                segnf_nop,
        .incore         = (size_t (*)(struct seg *, caddr_t, size_t, char *))
                segnf_nop,
        .lockop         = (int (*)(struct seg *, caddr_t, size_t, int, int,
            ulong_t *, size_t))segnf_nop,
        .getprot        = segnf_getprot,
        .getoffset      = segnf_getoffset,
        .gettype        = segnf_gettype,
        .getvp          = segnf_getvp,
        .advise         = (int (*)(struct seg *, caddr_t, size_t, uint_t))
                segnf_nop,
        .dump           = segnf_dump,
        .pagelock       = segnf_pagelock,
        .setpagesize    = segnf_setpagesize,
        .getmemid       = segnf_getmemid,
        .getpolicy      = segnf_getpolicy,
};
```

_____unchanged_portion_omitted_

```
/*ARGSUSED*/
static int
segnf_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp)
{
        return (ENODEV);
}

/*ARGSUSED*/
static lgrp_mem_policy_info_t *
segnf_getpolicy(struct seg *seg, caddr_t addr)
{
```

```
 492         return (NULL);
 483 }
```

_____***unchanged_portion_omitted_***