```
**********************************************************
  113437 Fri May  8 18:04:49 2015
new/usr/src/uts/common/vm/seg_dev.c
use NULL setpagesize segop as a shorthand for ENOTSUP
Instead of forcing every segment driver to implement a dummp function to
return (hopefully) ENOTSUP, handle NULL setpagesize segop function pointer
as "return ENOTSUP" shorthand.
**********************************************************
```

```
/*
 * 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
```

```
 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);
 184 static int      segdev_getmemid(struct seg *, caddr_t, memid_t *);

 186 /*
 187  * XXX  this struct is used by rootnex_map_fault to identify
 188  *      the segment it has been passed. So if you make it
 189  *      "static" you'll need to fix rootnex_map_fault.
```



```
 190  */
 191 struct seg_ops segdev_ops = {
 192         .dup            = segdev_dup,
 193         .unmap          = segdev_unmap,
 194         .free           = segdev_free,
 195         .fault          = segdev_fault,
 196         .faulta         = segdev_faulta,
 197         .setprot        = segdev_setprot,
 198         .checkprot      = segdev_checkprot,
 199         .kluster        = (int (*)())segdev_badop,
 200         .sync           = segdev_sync,
 201         .incore         = segdev_incore,
 202         .lockop         = segdev_lockop,
 203         .getprot        = segdev_getprot,
 204         .getoffset      = segdev_getoffset,
 205         .gettype        = segdev_gettype,
 206         .getvp          = segdev_getvp,
 207         .advise         = segdev_advise,
 208         .dump           = segdev_dump,
 209         .pagelock       = segdev_pagelock,
 211         .setpagesize    = segdev_setpagesize,
 210         .getmemid       = segdev_getmemid,
 211 };
______unchanged_portion_omitted_

2465 /*ARGSUSED*/
2466 static int
2467 segdev_pagelock(struct seg *seg, caddr_t addr, size_t len,
2468     struct page ***ppp, enum lock_type type, enum seg_rw rw)
2469 {
2470         TRACE_0(TR_FAC_DEVMAP, TR_DEVMAP_PAGELOCK,
2471             "segdev_pagelock:start");
2474         return (ENOTSUP);
2475 }

2477 /*ARGSUSED*/
2478 static int
2479 segdev_setpagesize(struct seg *seg, caddr_t addr, size_t len,
2480     uint_t szc)
2481 {
2472         return (ENOTSUP);
2473 }
______unchanged_portion_omitted_
```

```
**********************************************************
    9308 Fri May  8 18:04:49 2015
new/usr/src/uts/common/vm/seg_kpm.c
use NULL setpagesize segop as a shorthand for ENOTSUP
Instead of forcing every segment driver to implement a dummp function to
return (hopefully) ENOTSUP, handle NULL setpagesize segop function pointer
as "return ENOTSUP" shorthand.
**********************************************************
```

```
/*
 * CDDL HEADER START
 *
 * The contents of this file are subject to the terms of the
 * Common Development and Distribution License, Version 1.0 only
 * (the "License").  You may not use this file except in compliance
 * with the License.
 *
 * You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE
 * or http://www.opensolaris.org/os/licensing.
 * See the License for the specific language governing permissions
 * and limitations under the License.
 *
 * When distributing Covered Code, include this CDDL HEADER in each
 * file and include the License file at usr/src/OPENSOLARIS.LICENSE.
 * If applicable, add the following below this CDDL HEADER, with the
 * fields enclosed by brackets "[]" replaced with your own identifying
 * information: Portions Copyright [yyyy] [name of copyright owner]
 *
 * CDDL HEADER END
 */
/*
 * Copyright 2006 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 */

/*
 * Kernel Physical Mapping (kpm) segment driver (segkpm).
 *
 * This driver delivers along with the hat_kpm* interfaces an alternative
 * mechanism for kernel mappings within the 64-bit Solaris operating system,
 * which allows the mapping of all physical memory into the kernel address
 * space at once. This is feasible in 64 bit kernels, e.g. for Ultrasparc II
 * and beyond processors, since the available VA range is much larger than
 * possible physical memory. Momentarily all physical memory is supported,
 * that is represented by the list of memory segments (memsegs).
 *
 * Segkpm mappings have also very low overhead and large pages are used
 * (when possible) to minimize the TLB and TSB footprint. It is also
 * extentable for other than Sparc architectures (e.g. AMD64). Main
 * advantage is the avoidance of the TLB-shootdown X-calls, which are
 * normally needed when a kernel (global) mapping has to be removed.
 *
 * First example of a kernel facility that uses the segkpm mapping scheme
 * is seg_map, where it is used as an alternative to hat_memload().
 * See also hat layer for more information about the hat_kpm* routines.
 * The kpm facilty can be turned off at boot time (e.g. /etc/system).
 */

#include <sys/types.h>
#include <sys/param.h>
#include <sys/sysmacros.h>
#include <sys/systm.h>
#include <sys/vnode.h>
#include <sys/cmn_err.h>
#include <sys/debug.h>
#include <sys/thread.h>
#include <sys/cpuvar.h>
```



```
#include <sys/bitmap.h>
#include <sys/atomic.h>
#include <sys/lgrp.h>

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

/*
 * Global kpm controls.
 * See also platform and mmu specific controls.
 *
 * kpm_enable -- global on/off switch for segkpm.
 * . Set by default on 64bit platforms that have kpm support.
 * . Will be disabled from platform layer if not supported.
 * . Can be disabled via /etc/system.
 *
 * kpm_smallpages -- use only regular/system pagesize for kpm mappings.
 * . Can be useful for critical debugging of kpm clients.
 * . Set to zero by default for platforms that support kpm large pages.
 *   The use of kpm large pages reduces the footprint of kpm meta data
 *   and has all the other advantages of using large pages (e.g TLB
 *   miss reduction).
 * . Set by default for platforms that don't support kpm large pages or
 *   where large pages cannot be used for other reasons (e.g. there are
 *   only few full associative TLB entries available for large pages).
 *
 * segmap_kpm -- separate on/off switch for segmap using segkpm:
 * . Set by default.
 * . Will be disabled when kpm_enable is zero.
 * . Will be disabled when MAXBSIZE != PAGESIZE.
 * . Can be disabled via /etc/system.
 *
 */
int kpm_enable = 1;
int kpm_smallpages = 0;
int segmap_kpm = 1;

/*
 * Private seg op routines.
 */
faultcode_t segkpm_fault(struct hat *hat, struct seg *seg, caddr_t addr,
                         size_t len, enum fault_type type, enum seg_rw rw);
static void     segkpm_dump(struct seg *);
static int      segkpm_pagelock(struct seg *seg, caddr_t addr, size_t len,
                         struct page ***page, enum lock_type type,
                         enum seg_rw rw);

static struct seg_ops segkpm_ops = {
        .fault          = segkpm_fault,
        .dump           = segkpm_dump,
        .pagelock       = segkpm_pagelock,
//#ifndef SEGKPM_SUPPORT
#if 0
#error FIXME: define nop
        .dup            = nop,
        .unmap          = nop,
        .free           = nop,
        .faulta         = nop,
        .setprot        = nop,
        .checkprot      = nop,
        .kluster        = nop,
        .sync           = nop,
```

```
 125         .incore         = nop,
 126         .lockop         = nop,
 127         .getprot        = nop,
 128         .getoffset      = nop,
 129         .gettype        = nop,
 130         .getvp          = nop,
 131         .advise         = nop,
 132         .setpagesize    = nop,
 132         .getpolicy      = nop,
 133 #endif
 134 };
```

_____unchanged_portion_omitted_

```
**********************************************************
   82299 Fri May  8 18:04:49 2015
new/usr/src/uts/common/vm/seg_spt.c
use NULL setpagesize segop as a shorthand for ENOTSUP
Instead of forcing every segment driver to implement a dummp function to
return (hopefully) ENOTSUP, handle NULL setpagesize segop function pointer
as "return ENOTSUP" shorthand.
**********************************************************
______unchanged_portion_omitted_

  85 static int segspt_shmdup(struct seg *seg, struct seg *newseg);
  86 static int segspt_shmunmap(struct seg *seg, caddr_t raddr, size_t ssize);
  87 static void segspt_shmfree(struct seg *seg);
  88 static faultcode_t segspt_shmfault(struct hat *hat, struct seg *seg,
  89                 caddr_t addr, size_t len, enum fault_type type, enum seg_rw rw);
  90 static faultcode_t segspt_shmfaulta(struct seg *seg, caddr_t addr);
  91 static int segspt_shmsetprot(register struct seg *seg, register caddr_t addr,
  92                         register size_t len, register uint_t prot);
  93 static int segspt_shmcheckprot(struct seg *seg, caddr_t addr, size_t size,
  94                         uint_t prot);
  95 static int      segspt_shmkluster(struct seg *seg, caddr_t addr, ssize_t delta);
  96 static size_t segspt_shmincore(struct seg *seg, caddr_t addr, size_t len,
  97                         register char *vec);
  98 static int segspt_shmsync(struct seg *seg, register caddr_t addr, size_t len,
  99                         int attr, uint_t flags);
 100 static int segspt_shmlockop(struct seg *seg, caddr_t addr, size_t len,
 101                         int attr, int op, ulong_t *lockmap, size_t pos);
 102 static int segspt_shmgetprot(struct seg *seg, caddr_t addr, size_t len,
 103                         uint_t *protv);
 104 static u_offset_t segspt_shmgetoffset(struct seg *seg, caddr_t addr);
 105 static int segspt_shmgettype(struct seg *seg, caddr_t addr);
 106 static int segspt_shmgetvp(struct seg *seg, caddr_t addr, struct vnode **vpp);
 107 static int segspt_shmadvise(struct seg *seg, caddr_t addr, size_t len,
 108                         uint_t behav);
 109 static void segspt_shmdump(struct seg *seg);
 110 static int segspt_shmpagelock(struct seg *, caddr_t, size_t,
 111                         struct page ***, enum lock_type, enum seg_rw);
 112 static int segspt_shmsetpgsz(struct seg *, caddr_t, size_t, uint_t);
 112 static int segspt_shmgetmemid(struct seg *, caddr_t, memid_t *);
 113 static lgrp_mem_policy_info_t *segspt_shmgetpolicy(struct seg *, caddr_t);

 115 struct seg_ops segspt_shmops = {
 116         .dup            = segspt_shmdup,
 117         .unmap          = segspt_shmunmap,
 118         .free           = segspt_shmfree,
 119         .fault          = segspt_shmfault,
 120         .faulta         = segspt_shmfaulta,
 121         .setprot        = segspt_shmsetprot,
 122         .checkprot      = segspt_shmcheckprot,
 123         .kluster        = segspt_shmkluster,
 124         .sync           = segspt_shmsync,
 125         .incore         = segspt_shmincore,
 126         .lockop         = segspt_shmlockop,
 127         .getprot        = segspt_shmgetprot,
 128         .getoffset      = segspt_shmgetoffset,
 129         .gettype        = segspt_shmgettype,
 130         .getvp          = segspt_shmgetvp,
 131         .advise         = segspt_shmadvise,
 132         .dump           = segspt_shmdump,
 133         .pagelock       = segspt_shmpagelock,
 135         .setpagesize    = segspt_shmsetpgsz,
 134         .getmemid       = segspt_shmgetmemid,
 135         .getpolicy      = segspt_shmgetpolicy,
 136 };
______unchanged_portion_omitted_
```



```
2952 /*ARGSUSED*/
2953 void
2954 segspt_shmdump(struct seg *seg)
2955 {
2956         /* no-op for ISM segment */
2959 }

2961 /*ARGSUSED*/
2962 static faultcode_t
2963 segspt_shmsetpgsz(struct seg *seg, caddr_t addr, size_t len, uint_t szc)
2964 {
2965         return (ENOTSUP);
2957 }
______unchanged_portion_omitted_
```

****
55144 Fri May 8 18:04:49 2015
new/usr/src/uts/common/vm/vm_seg.c
use NULL setpagesize segop as a shorthand for ENOTSUP
Instead of forcing every segment driver to implement a dummp function to
return (hopefully) ENOTSUP, handle NULL setpagesize segop function pointer
as "return ENOTSUP" shorthand.
****
_____***unchanged_portion_omitted_***

```
2006 int
2007 segop_setpagesize(struct seg *seg, caddr_t addr, size_t len, uint_t szc)
2008 {
2009         if (seg->s_ops->setpagesize == NULL)
2010                 return (ENOTSUP);
2009         VERIFY3P(seg->s_ops->setpagesize, !=, NULL);

2012         return (seg->s_ops->setpagesize(seg, addr, len, szc));
2013 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   16536 Fri May  8 18:04:50 2015
new/usr/src/uts/i86xpv/vm/seg_mf.c
use NULL setpagesize segop as a shorthand for ENOTSUP
Instead of forcing every segment driver to implement a dummp function to
return (hopefully) ENOTSUP, handle NULL setpagesize segop function pointer
as "return ENOTSUP" shorthand.
**********************************************************
______unchanged_portion_omitted_

 485 /*ARGSUSED*/
 486 static int
 487 segmf_setpagesize(struct seg *seg, caddr_t addr, size_t len, uint_t szc)
 488 {
 489         return (ENOTSUP);
 490 }

 485 static int
 486 segmf_getmemid(struct seg *seg, caddr_t addr, memid_t *memid)
 487 {
 488         struct segmf_data *data = seg->s_data;

 490         memid->val[0] = (uintptr_t)VTOCVP(data->vp);
 491         memid->val[1] = (uintptr_t)seg_page(seg, addr);
 492         return (0);
 493 }
______unchanged_portion_omitted_

 739 static struct seg_ops segmf_ops = {
 740         .dup            = segmf_dup,
 741         .unmap          = segmf_unmap,
 742         .free           = segmf_free,
 743         .fault          = segmf_fault,
 744         .faulta         = segmf_faulta,
 745         .setprot        = segmf_setprot,
 746         .checkprot      = segmf_checkprot,
 747         .kluster        = segmf_kluster,
 748         .sync           = segmf_sync,
 749         .incore         = segmf_incore,
 750         .lockop         = segmf_lockop,
 751         .getprot        = segmf_getprot,
 752         .getoffset      = segmf_getoffset,
 753         .gettype        = segmf_gettype,
 754         .getvp          = segmf_getvp,
 755         .advise         = segmf_advise,
 756         .dump           = segmf_dump,
 757         .pagelock       = segmf_pagelock,
 765         .setpagesize    = segmf_setpagesize,
 758         .getmemid       = segmf_getmemid,
 759 };
______unchanged_portion_omitted_
```

```
**********************************************************
   11638 Fri May  8 18:04:50 2015
new/usr/src/uts/sparc/v9/vm/seg_nf.c
use NULL setpagesize segop as a shorthand for ENOTSUP
Instead of forcing every segment driver to implement a dummp function to
return (hopefully) ENOTSUP, handle NULL setpagesize segop function pointer
as "return ENOTSUP" shorthand.
**********************************************************
```

```
/*
 * 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 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);


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,
        .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,
};
_____unchanged_portion_omitted_

/*
 * segnf pages are not dumped, so we just return
 */
/* ARGSUSED */
static void
segnf_dump(struct seg *seg)
{}

/*ARGSUSED*/
static int
segnf_pagelock(struct seg *seg, caddr_t addr, size_t len,
    struct page ***ppp, enum lock_type type, enum seg_rw rw)
{
        return (ENOTSUP);
}
```

```
459 /*ARGSUSED*/
460 static int
461 segnf_setpagesize(struct seg *seg, caddr_t addr, size_t len,
462     uint_t szc)
463 {
464         return (ENOTSUP);
465 }
```

_____***unchanged_portion_omitted_***