```
**********************************************************
   36495 Tue Nov 24 09:35:03 2015
new/usr/src/uts/common/vm/seg_kp.c
6150 use NULL getmemid segop as a shorthand for ENODEV
**********************************************************
```

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

  98 /*
  99  * Lock used to protect the hash table(s) and caches.
 100  */
 101 static kmutex_t segkp_lock;

 103 /*
 104  * The segkp caches
 105  */
 106 static struct segkp_cache segkp_cache[SEGKP_MAX_CACHE];

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

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

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

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

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

 134 hrtime_t        red_deep_hires;
 135 kthread_t       *red_deep_thread;

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

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

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

1386 /*ARGSUSED*/
1387 static int
1388 segkp_pagelock(struct seg *seg, caddr_t addr, size_t len,
1389     struct page ***ppp, enum lock_type type, enum seg_rw rw)
1390 {
1391         return (ENOTSUP);
1394 }

1396 /*ARGSUSED*/
1397 static int
1398 segkp_getmemid(struct seg *seg, caddr_t addr, memid_t *memidp)
1399 {
1400         return (ENODEV);
1392 }
______unchanged_portion_omitted_
```

****
94463 Tue Nov 24 09:35:03 2015
new/usr/src/uts/common/vm/vm_as.c
6150 use NULL getmemid segop as a shorthand for ENODEV
****
_____***unchanged_portion_omitted_***

```
3660 /*
3661  * return memory object ID
3662  */
3663 int
3664 as_getmemid(struct as *as, caddr_t addr, memid_t *memidp)
3665 {
3666         struct seg      *seg;
3667         int             sts;

3669         AS_LOCK_ENTER(as, &as->a_lock, RW_READER);
3670         seg = as_segat(as, addr);
3671         if (seg == NULL) {
3672                 AS_LOCK_EXIT(as, &as->a_lock);
3673                 return (EFAULT);
3674         }
3675         /*
3676          * catch old drivers which may not support getmemid
3677          */
3678         if (seg->s_ops->getmemid == NULL) {
3679                 AS_LOCK_EXIT(as, &as->a_lock);
3680                 return (ENODEV);
3681         }

3676         sts = segop_getmemid(seg, addr, memidp);

3678         AS_LOCK_EXIT(as, &as->a_lock);
3679         return (sts);
3680 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   54477 Tue Nov 24 09:35:04 2015
new/usr/src/uts/common/vm/vm_seg.c
6150 use NULL getmemid segop as a shorthand for ENODEV
**********************************************************
______unchanged_portion_omitted_

1983 int
1984 segop_getmemid(struct seg *seg, caddr_t addr, memid_t *mp)
1985 {
1986         if (seg->s_ops->getmemid == NULL)
1987                 return (ENODEV);

1989 #endif /* ! codereview */
1990         return (seg->s_ops->getmemid(seg, addr, mp));
1991 }

1993 struct lgrp_mem_policy_info *
1994 segop_getpolicy(struct seg *seg, caddr_t addr)
1995 {
1996         if (seg->s_ops->getpolicy == NULL)
1997                 return (NULL);

1999         return (seg->s_ops->getpolicy(seg, addr));
2000 }

2002 int
2003 segop_capable(struct seg *seg, segcapability_t cap)
2004 {
2005         if (seg->s_ops->capable == NULL)
2006                 return (0);

2008         return (seg->s_ops->capable(seg, cap));
2009 }

2011 int
2012 segop_inherit(struct seg *seg, caddr_t addr, size_t len, uint_t op)
2013 {
2014         if (seg->s_ops->inherit == NULL)
2015                 return (ENOTSUP);

2017         return (seg->s_ops->inherit(seg, addr, len, op));
2018 }
```

```
**********************************************************
   12030 Tue Nov 24 09:35:04 2015
new/usr/src/uts/sparc/v9/vm/seg_nf.c
6150 use NULL getmemid segop as a shorthand for ENODEV
**********************************************************
```

```
/*
 * 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);


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,
};
```

_____**unchanged_portion_omitted_**

```
/*ARGSUSED*/
static int
segnf_setpagesize(struct seg *seg, caddr_t addr, size_t len,
    uint_t szc)
{
        return (ENOTSUP);
}

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

_____**unchanged_portion_omitted_**