```
**********************************************************
  114109 Tue Nov 24 09:34:38 2015
new/usr/src/uts/common/vm/seg_dev.c
6144 use C99 initializers in segment ops structures
**********************************************************
/*
 * 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.
 * We can also use an anonymous memory page as there is no requirement to
 * keep the page locked, however this complicates the fault code. RFE.
 */
static struct vnode trashvp;
static struct page *trashpp;

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

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

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

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

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

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

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

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

/*
 * Private seg op routines.
 */
static int      segdev_dup(struct seg *, struct seg *);
static int      segdev_unmap(struct seg *, caddr_t, size_t);
static void     segdev_free(struct seg *);
static faultcode_t segdev_fault(struct hat *, struct seg *, caddr_t, size_t,
                    enum fault_type, enum seg_rw);
static faultcode_t segdev_faulta(struct seg *, caddr_t);
static int      segdev_setprot(struct seg *, caddr_t, size_t, uint_t);
static int      segdev_checkprot(struct seg *, caddr_t, size_t, uint_t);
static void     segdev_badop(void);
static int      segdev_sync(struct seg *, caddr_t, size_t, int, uint_t);
static size_t   segdev_incore(struct seg *, caddr_t, size_t, char *);
static int      segdev_lockop(struct seg *, caddr_t, size_t, int, int,
                    ulong_t *, size_t);
static int      segdev_getprot(struct seg *, caddr_t, size_t, uint_t *);
static u_offset_t       segdev_getoffset(struct seg *, caddr_t);
static int      segdev_gettype(struct seg *, caddr_t);
static int      segdev_getvp(struct seg *, caddr_t, struct vnode **);
static int      segdev_advise(struct seg *, caddr_t, size_t, uint_t);
static void     segdev_dump(struct seg *);
static int      segdev_pagelock(struct seg *, caddr_t, size_t,
                    struct page ***, enum lock_type, enum seg_rw);
static int      segdev_setpagesize(struct seg *, caddr_t, size_t, uint_t);
static int      segdev_getmemid(struct seg *, caddr_t, memid_t *);
static lgrp_mem_policy_info_t   *segdev_getpolicy(struct seg *, caddr_t);
static int      segdev_capable(struct seg *, segcapability_t);

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

```
struct seg_ops segdev_ops = {
        .dup            = segdev_dup,
        .unmap          = segdev_unmap,
        .free           = segdev_free,
        .fault          = segdev_fault,
        .faulta         = segdev_faulta,
        .setprot        = segdev_setprot,
        .checkprot      = segdev_checkprot,
        .kluster        = (int (*)())segdev_badop,
        .sync           = segdev_sync,
        .incore         = segdev_incore,
        .lockop         = segdev_lockop,
        .getprot        = segdev_getprot,
        .getoffset      = segdev_getoffset,
        .gettype        = segdev_gettype,
        .getvp          = segdev_getvp,
        .advise         = segdev_advise,
        .dump           = segdev_dump,
        .pagelock       = segdev_pagelock,
        .setpagesize    = segdev_setpagesize,
        .getmemid       = segdev_getmemid,
        .getpolicy      = segdev_getpolicy,
        .capable        = segdev_capable,
        .inherit        = seg_inherit_notsup,
        segdev_dup,
        segdev_unmap,
        segdev_free,
        segdev_fault,
        segdev_faulta,
        segdev_setprot,
        segdev_checkprot,
        (int (*)())segdev_badop,        /* kluster */
        (size_t (*)(struct seg *))NULL, /* swapout */
        segdev_sync,                    /* sync */
        segdev_incore,
        segdev_lockop,                  /* lockop */
        segdev_getprot,
        segdev_getoffset,
        segdev_gettype,
        segdev_getvp,
        segdev_advise,
        segdev_dump,
        segdev_pagelock,
        segdev_setpagesize,
        segdev_getmemid,
        segdev_getpolicy,
        segdev_capable,
        seg_inherit_notsup
};
```

______unchanged_portion_omitted_

```
**********************************************************
   45463 Tue Nov 24 09:34:38 2015
new/usr/src/uts/common/vm/seg_kmem.c
6144 use C99 initializers in segment ops structures
**********************************************************
______unchanged_portion_omitted_

static struct seg_ops segkmem_ops = {
        .dup            = SEGKMEM_BADOP(int),
        .unmap          = SEGKMEM_BADOP(int),
        .free           = SEGKMEM_BADOP(void),
        .fault          = segkmem_fault,
        .faulta         = SEGKMEM_BADOP(faultcode_t),
        .setprot        = segkmem_setprot,
        .checkprot      = segkmem_checkprot,
        .kluster        = segkmem_kluster,
        .swapout        = SEGKMEM_BADOP(size_t),
        .sync           = SEGKMEM_BADOP(int),
        .incore         = SEGKMEM_BADOP(size_t),
        .lockop         = SEGKMEM_BADOP(int),
        .getprot        = SEGKMEM_BADOP(int),
        .getoffset      = SEGKMEM_BADOP(u_offset_t),
        .gettype        = SEGKMEM_BADOP(int),
        .getvp          = SEGKMEM_BADOP(int),
        .advise         = SEGKMEM_BADOP(int),
        .dump           = segkmem_dump,
        .pagelock       = segkmem_pagelock,
        .setpagesize    = SEGKMEM_BADOP(int),
        .getmemid       = segkmem_getmemid,
        .getpolicy      = segkmem_getpolicy,
        .capable        = segkmem_capable,
        .inherit        = seg_inherit_notsup,
        SEGKMEM_BADOP(int),             /* dup */
        SEGKMEM_BADOP(int),             /* unmap */
        SEGKMEM_BADOP(void),            /* free */
        segkmem_fault,
        SEGKMEM_BADOP(faultcode_t),     /* faulta */
        segkmem_setprot,
        segkmem_checkprot,
        segkmem_kluster,
        SEGKMEM_BADOP(size_t),          /* swapout */
        SEGKMEM_BADOP(int),             /* sync */
        SEGKMEM_BADOP(size_t),          /* incore */
        SEGKMEM_BADOP(int),             /* lockop */
        SEGKMEM_BADOP(int),             /* getprot */
        SEGKMEM_BADOP(u_offset_t),      /* getoffset */
        SEGKMEM_BADOP(int),             /* gettype */
        SEGKMEM_BADOP(int),             /* getvp */
        SEGKMEM_BADOP(int),             /* advise */
        segkmem_dump,
        segkmem_pagelock,
        SEGKMEM_BADOP(int),             /* setpgsz */
        segkmem_getmemid,
        segkmem_getpolicy,              /* getpolicy */
        segkmem_capable,                /* capable */
        seg_inherit_notsup              /* inherit */
};
______unchanged_portion_omitted_
```

```
**********************************************************
   37167 Tue Nov 24 09:34:38 2015
new/usr/src/uts/common/vm/seg_kp.c
6144 use C99 initializers in segment ops structures
**********************************************************
/*
 * 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;

/*
```

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

hrtime_t        red_deep_hires;
kthread_t       *red_deep_thread;

uint32_t        red_nmapped;
uint32_t        red_closest = UINT_MAX;
uint32_t        red_ndoubles;

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

static struct   seg_ops segkp_ops = {
        .dup            = SEGKP_BADOP(int),
        .unmap          = SEGKP_BADOP(int),
        .free           = SEGKP_BADOP(void),
        .fault          = segkp_fault,
        .faulta         = SEGKP_BADOP(faultcode_t),
        .setprot        = SEGKP_BADOP(int),
        .checkprot      = segkp_checkprot,
        .kluster        = segkp_kluster,
        .swapout        = SEGKP_BADOP(size_t),
        .sync           = SEGKP_BADOP(int),
        .incore         = SEGKP_BADOP(size_t),
        .lockop         = SEGKP_BADOP(int),
        .getprot        = SEGKP_BADOP(int),
        .getoffset      = SEGKP_BADOP(u_offset_t),
        .gettype        = SEGKP_BADOP(int),
        .getvp          = SEGKP_BADOP(int),
        .advise         = SEGKP_BADOP(int),
        .dump           = segkp_dump,
        .pagelock       = segkp_pagelock,
        .setpagesize    = SEGKP_BADOP(int),
        .getmemid       = segkp_getmemid,
        .getpolicy      = segkp_getpolicy,
        .capable        = segkp_capable,
        .inherit        = seg_inherit_notsup,
        SEGKP_BADOP(int),               /* dup */
        SEGKP_BADOP(int),               /* unmap */
        SEGKP_BADOP(void),              /* free */
        segkp_fault,
        SEGKP_BADOP(faultcode_t),       /* faulta */
        SEGKP_BADOP(int),               /* setprot */
        segkp_checkprot,
        segkp_kluster,
        SEGKP_BADOP(size_t),            /* swapout */
        SEGKP_BADOP(int),               /* sync */
        SEGKP_BADOP(size_t),            /* incore */
        SEGKP_BADOP(int),               /* lockop */
        SEGKP_BADOP(int),               /* getprot */
        SEGKP_BADOP(u_offset_t),                /* getoffset */
        SEGKP_BADOP(int),               /* gettype */
        SEGKP_BADOP(int),               /* getvp */
        SEGKP_BADOP(int),               /* advise */
        segkp_dump,                     /* dump */
        segkp_pagelock,                 /* pagelock */
        SEGKP_BADOP(int),               /* setpgsz */
        segkp_getmemid,                 /* getmemid */
```

```
        segkp_getpolicy,                /* getpolicy */
        segkp_capable,                  /* capable */
        seg_inherit_notsup              /* inherit */
};
```

______***unchanged_portion_omitted_***

```
**********************************************************
    9872 Tue Nov 24 09:34:38 2015
new/usr/src/uts/common/vm/seg_kpm.c
6144 use C99 initializers in segment ops structures
**********************************************************
/*
 * 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 void     segkpm_badop(void);
static int      segkpm_notsup(void);
static int      segkpm_capable(struct seg *, segcapability_t);

#define SEGKPM_BADOP(t) (t(*)())segkpm_badop
#define SEGKPM_NOTSUP   (int(*)())segkpm_notsup

static struct seg_ops segkpm_ops = {
        .dup            = SEGKPM_BADOP(int),
        .unmap          = SEGKPM_BADOP(int),
        .free           = SEGKPM_BADOP(void),
        .fault          = segkpm_fault,
        .faulta         = SEGKPM_BADOP(int),
        .setprot        = SEGKPM_BADOP(int),
        .checkprot      = SEGKPM_BADOP(int),
        .kluster        = SEGKPM_BADOP(int),
        .swapout        = SEGKPM_BADOP(size_t),
        .sync           = SEGKPM_BADOP(int),
        .incore         = SEGKPM_BADOP(size_t),
        .lockop         = SEGKPM_BADOP(int),
        .getprot        = SEGKPM_BADOP(int),
        .getoffset      = SEGKPM_BADOP(u_offset_t),
        .gettype        = SEGKPM_BADOP(int),
        .getvp          = SEGKPM_BADOP(int),
        .advise         = SEGKPM_BADOP(int),
        .dump           = segkpm_dump,
        .pagelock       = SEGKPM_NOTSUP,
        .setpagesize    = SEGKPM_BADOP(int),
        .getmemid       = SEGKPM_BADOP(int),
        .getpolicy      = SEGKPM_BADOP(lgrp_mem_policy_info_t *),
        .capable        = segkpm_capable,
        .inherit        = seg_inherit_notsup,
        SEGKPM_BADOP(int),      /* dup */
        SEGKPM_BADOP(int),      /* unmap */
        SEGKPM_BADOP(void),     /* free */
        segkpm_fault,
        SEGKPM_BADOP(int),      /* faulta */
        SEGKPM_BADOP(int),      /* setprot */
        SEGKPM_BADOP(int),      /* checkprot */
        SEGKPM_BADOP(int),      /* kluster */
        SEGKPM_BADOP(size_t),   /* swapout */
        SEGKPM_BADOP(int),      /* sync */
        SEGKPM_BADOP(size_t),   /* incore */
        SEGKPM_BADOP(int),      /* lockop */
        SEGKPM_BADOP(int),      /* getprot */
        SEGKPM_BADOP(u_offset_t), /* getoffset */
        SEGKPM_BADOP(int),      /* gettype */
        SEGKPM_BADOP(int),      /* getvp */
        SEGKPM_BADOP(int),      /* advise */
        segkpm_dump,            /* dump */
        SEGKPM_NOTSUP,          /* pagelock */
        SEGKPM_BADOP(int),      /* setpgsz */
        SEGKPM_BADOP(int),      /* getmemid */
        SEGKPM_BADOP(lgrp_mem_policy_info_t *), /* getpolicy */
        segkpm_capable,         /* capable */
        seg_inherit_notsup      /* inherit */
};
```
______***unchanged_portion_omitted_***

```
**********************************************************
   58162 Tue Nov 24 09:34:39 2015
new/usr/src/uts/common/vm/seg_map.c
6144 use C99 initializers in segment ops structures
**********************************************************
/*
 * 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,
        .inherit        = seg_inherit_notsup,
        SEGMAP_BADOP(int),      /* dup */
        SEGMAP_BADOP(int),      /* unmap */
        segmap_free,
        segmap_fault,
        segmap_faulta,
        SEGMAP_BADOP(int),      /* setprot */
        segmap_checkprot,
        segmap_kluster,
        SEGMAP_BADOP(size_t),   /* swapout */
        SEGMAP_BADOP(int),      /* sync */
        SEGMAP_BADOP(size_t),   /* incore */
        SEGMAP_BADOP(int),      /* lockop */
        segmap_getprot,
        segmap_getoffset,
        segmap_gettype,
        segmap_getvp,
        SEGMAP_BADOP(int),      /* advise */
        segmap_dump,
        segmap_pagelock,        /* pagelock */
        SEGMAP_BADOP(int),      /* setpgsz */
        segmap_getmemid,        /* getmemid */
        segmap_getpolicy,       /* getpolicy */
        segmap_capable,         /* capable */
        seg_inherit_notsup      /* inherit */
};
```
______***unchanged_portion_omitted_***

```
**********************************************************
   84180 Tue Nov 24 09:34:39 2015
new/usr/src/uts/common/vm/seg_spt.c
6144 use C99 initializers in segment ops structures
**********************************************************
______unchanged_portion_omitted_

#define SEGSPT_BADOP(t) (t(*)())segspt_badop

struct seg_ops segspt_ops = {
        .dup            = SEGSPT_BADOP(int),
        .unmap          = segspt_unmap,
        .free           = segspt_free,
        .fault          = SEGSPT_BADOP(int),
        .faulta         = SEGSPT_BADOP(faultcode_t),
        .setprot        = SEGSPT_BADOP(int),
        .checkprot      = SEGSPT_BADOP(int),
        .kluster        = SEGSPT_BADOP(int),
        .swapout        = SEGSPT_BADOP(size_t),
        .sync           = SEGSPT_BADOP(int),
        .incore         = SEGSPT_BADOP(size_t),
        .lockop         = SEGSPT_BADOP(int),
        .getprot        = SEGSPT_BADOP(int),
        .getoffset      = SEGSPT_BADOP(u_offset_t),
        .gettype        = SEGSPT_BADOP(int),
        .getvp          = SEGSPT_BADOP(int),
        .advise         = SEGSPT_BADOP(int),
        .dump           = SEGSPT_BADOP(void),
        .pagelock       = SEGSPT_BADOP(int),
        .setpagesize    = SEGSPT_BADOP(int),
        .getmemid       = SEGSPT_BADOP(int),
        .getpolicy      = segspt_getpolicy,
        .capable        = SEGSPT_BADOP(int),
        .inherit        = seg_inherit_notsup,
        SEGSPT_BADOP(int),              /* dup */
        segspt_unmap,
        segspt_free,
        SEGSPT_BADOP(int),              /* fault */
        SEGSPT_BADOP(faultcode_t),      /* faulta */
        SEGSPT_BADOP(int),              /* setprot */
        SEGSPT_BADOP(int),              /* checkprot */
        SEGSPT_BADOP(int),              /* kluster */
        SEGSPT_BADOP(size_t),           /* swapout */
        SEGSPT_BADOP(int),              /* sync */
        SEGSPT_BADOP(size_t),           /* incore */
        SEGSPT_BADOP(int),              /* lockop */
        SEGSPT_BADOP(int),              /* getprot */
        SEGSPT_BADOP(u_offset_t),       /* getoffset */
        SEGSPT_BADOP(int),              /* gettype */
        SEGSPT_BADOP(int),              /* getvp */
        SEGSPT_BADOP(int),              /* advise */
        SEGSPT_BADOP(void),             /* dump */
        SEGSPT_BADOP(int),              /* pagelock */
        SEGSPT_BADOP(int),              /* setpgsz */
        SEGSPT_BADOP(int),              /* getmemid */
        segspt_getpolicy,               /* getpolicy */
        SEGSPT_BADOP(int),              /* capable */
        seg_inherit_notsup              /* inherit */
};

static int segspt_shmdup(struct seg *seg, struct seg *newseg);
static int segspt_shmunmap(struct seg *seg, caddr_t raddr, size_t ssize);
static void segspt_shmfree(struct seg *seg);
static faultcode_t segspt_shmfault(struct hat *hat, struct seg *seg,
                caddr_t addr, size_t len, enum fault_type type, enum seg_rw rw);
static faultcode_t segspt_shmfaulta(struct seg *seg, caddr_t addr);
```

```
static int segspt_shmsetprot(register struct seg *seg, register caddr_t addr,
                        register size_t len, register uint_t prot);
static int segspt_shmcheckprot(struct seg *seg, caddr_t addr, size_t size,
                        uint_t prot);
static int      segspt_shmkluster(struct seg *seg, caddr_t addr, ssize_t delta);
static size_t   segspt_shmswapout(struct seg *seg);
static size_t segspt_shmincore(struct seg *seg, caddr_t addr, size_t len,
                        register char *vec);
static int segspt_shmsync(struct seg *seg, register caddr_t addr, size_t len,
                        int attr, uint_t flags);
static int segspt_shmlockop(struct seg *seg, caddr_t addr, size_t len,
                        int attr, int op, ulong_t *lockmap, size_t pos);
static int segspt_shmgetprot(struct seg *seg, caddr_t addr, size_t len,
                        uint_t *protv);
static u_offset_t segspt_shmgetoffset(struct seg *seg, caddr_t addr);
static int segspt_shmgettype(struct seg *seg, caddr_t addr);
static int segspt_shmgetvp(struct seg *seg, caddr_t addr, struct vnode **vpp);
static int segspt_shmadvise(struct seg *seg, caddr_t addr, size_t len,
                        uint_t behav);
static void segspt_shmdump(struct seg *seg);
static int segspt_shmpagelock(struct seg *, caddr_t, size_t,
                        struct page ***, enum lock_type, enum seg_rw);
static int segspt_shmsetpgsz(struct seg *, caddr_t, size_t, uint_t);
static int segspt_shmgetmemid(struct seg *, caddr_t, memid_t *);
static lgrp_mem_policy_info_t *segspt_shmgetpolicy(struct seg *, caddr_t);
static int segspt_shmcapable(struct seg *, segcapability_t);

struct seg_ops segspt_shmops = {
        .dup            = segspt_shmdup,
        .unmap          = segspt_shmunmap,
        .free           = segspt_shmfree,
        .fault          = segspt_shmfault,
        .faulta         = segspt_shmfaulta,
        .setprot        = segspt_shmsetprot,
        .checkprot      = segspt_shmcheckprot,
        .kluster        = segspt_shmkluster,
        .swapout        = segspt_shmswapout,
        .sync           = segspt_shmsync,
        .incore         = segspt_shmincore,
        .lockop         = segspt_shmlockop,
        .getprot        = segspt_shmgetprot,
        .getoffset      = segspt_shmgetoffset,
        .gettype        = segspt_shmgettype,
        .getvp          = segspt_shmgetvp,
        .advise         = segspt_shmadvise,
        .dump           = segspt_shmdump,
        .pagelock       = segspt_shmpagelock,
        .setpagesize    = segspt_shmsetpgsz,
        .getmemid       = segspt_shmgetmemid,
        .getpolicy      = segspt_shmgetpolicy,
        .capable        = segspt_shmcapable,
        .inherit        = seg_inherit_notsup,
        segspt_shmdup,
        segspt_shmunmap,
        segspt_shmfree,
        segspt_shmfault,
        segspt_shmfaulta,
        segspt_shmsetprot,
        segspt_shmcheckprot,
        segspt_shmkluster,
        segspt_shmswapout,
        segspt_shmsync,
        segspt_shmincore,
        segspt_shmlockop,
        segspt_shmgetprot,
        segspt_shmgetoffset,
        segspt_shmgettype,
        segspt_shmgetvp,
        segspt_shmadvise,       /* advise */
        segspt_shmdump,
        segspt_shmpagelock,
        segspt_shmsetpgsz,
        segspt_shmgetmemid,
        segspt_shmgetpolicy,
        segspt_shmcapable,
        seg_inherit_notsup
};
```
______***unchanged_portion_omitted_***

```
**********************************************************
  286002 Tue Nov 24 09:34:39 2015
new/usr/src/uts/common/vm/seg_vn.c
6144 use C99 initializers in segment ops structures
**********************************************************
/*
 * 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) 1986, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright 2015, Joyent, Inc. All rights reserved.
 * Copyright 2015 Nexenta Systems, Inc.  All rights reserved.
 */

/*      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 - shared or copy-on-write from a vnode/anonymous memory.
 */

#include <sys/types.h>
#include <sys/param.h>
#include <sys/t_lock.h>
#include <sys/errno.h>
#include <sys/systm.h>
#include <sys/mman.h>
#include <sys/debug.h>
#include <sys/cred.h>
#include <sys/vmsystm.h>
#include <sys/tuneable.h>
#include <sys/bitmap.h>
#include <sys/swap.h>
#include <sys/kmem.h>
#include <sys/sysmacros.h>
#include <sys/vtrace.h>
#include <sys/cmn_err.h>
#include <sys/callb.h>
#include <sys/vm.h>
```

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

#include <vm/hat.h>
#include <vm/as.h>
#include <vm/seg.h>
#include <vm/seg_vn.h>
#include <vm/pvn.h>
#include <vm/anon.h>
#include <vm/page.h>
#include <vm/vpage.h>
#include <sys/proc.h>
#include <sys/task.h>
#include <sys/project.h>
#include <sys/zone.h>
#include <sys/shm_impl.h>

/*
 * segvn_fault needs a temporary page list array.  To avoid calling kmem all
 * the time, it creates a small (PVN_GETPAGE_NUM entry) array and uses it if
 * it can.  In the rare case when this page list is not large enough, it
 * goes and gets a large enough array from kmem.
 *
 * This small page list array covers either 8 pages or 64kB worth of pages -
 * whichever is smaller.
 */
#define PVN_MAX_GETPAGE_SZ      0x10000
#define PVN_MAX_GETPAGE_NUM     0x8

#if PVN_MAX_GETPAGE_SZ > PVN_MAX_GETPAGE_NUM * PAGESIZE
#define PVN_GETPAGE_SZ  ptob(PVN_MAX_GETPAGE_NUM)
#define PVN_GETPAGE_NUM PVN_MAX_GETPAGE_NUM
#else
#define PVN_GETPAGE_SZ  PVN_MAX_GETPAGE_SZ
#define PVN_GETPAGE_NUM btop(PVN_MAX_GETPAGE_SZ)
#endif

/*
 * Private seg op routines.
 */
static int      segvn_dup(struct seg *seg, struct seg *newseg);
static int      segvn_unmap(struct seg *seg, caddr_t addr, size_t len);
static void     segvn_free(struct seg *seg);
static faultcode_t segvn_fault(struct hat *hat, struct seg *seg,
                    caddr_t addr, size_t len, enum fault_type type,
                    enum seg_rw rw);
static faultcode_t segvn_faulta(struct seg *seg, caddr_t addr);
static int      segvn_setprot(struct seg *seg, caddr_t addr,
                    size_t len, uint_t prot);
static int      segvn_checkprot(struct seg *seg, caddr_t addr,
                    size_t len, uint_t prot);
static int      segvn_kluster(struct seg *seg, caddr_t addr, ssize_t delta);
static size_t   segvn_swapout(struct seg *seg);
static int      segvn_sync(struct seg *seg, caddr_t addr, size_t len,
                    int attr, uint_t flags);
static size_t   segvn_incore(struct seg *seg, caddr_t addr, size_t len,
                    char *vec);
static int      segvn_lockop(struct seg *seg, caddr_t addr, size_t len,
                    int attr, int op, ulong_t *lockmap, size_t pos);
static int      segvn_getprot(struct seg *seg, caddr_t addr, size_t len,
                    uint_t *protv);
static u_offset_t       segvn_getoffset(struct seg *seg, caddr_t addr);
static int      segvn_gettype(struct seg *seg, caddr_t addr);
static int      segvn_getvp(struct seg *seg, caddr_t addr,
                    struct vnode **vpp);
static int      segvn_advise(struct seg *seg, caddr_t addr, size_t len,
                    uint_t behav);
static void     segvn_dump(struct seg *seg);
static int      segvn_pagelock(struct seg *seg, caddr_t addr, size_t len,
                    struct page ***ppp, enum lock_type type, enum seg_rw rw);
static int      segvn_setpagesize(struct seg *seg, caddr_t addr, size_t len,
                    uint_t szc);
static int      segvn_getmemid(struct seg *seg, caddr_t addr,
                    memid_t *memidp);
static lgrp_mem_policy_info_t   *segvn_getpolicy(struct seg *, caddr_t);
static int      segvn_capable(struct seg *seg, segcapability_t capable);
static int      segvn_inherit(struct seg *, caddr_t, size_t, uint_t);

struct  seg_ops segvn_ops = {
        .dup            = segvn_dup,
        .unmap          = segvn_unmap,
        .free           = segvn_free,
        .fault          = segvn_fault,
        .faulta         = segvn_faulta,
        .setprot        = segvn_setprot,
        .checkprot      = segvn_checkprot,
        .kluster        = segvn_kluster,
        .swapout        = segvn_swapout,
        .sync           = segvn_sync,
        .incore         = segvn_incore,
        .lockop         = segvn_lockop,
        .getprot        = segvn_getprot,
        .getoffset      = segvn_getoffset,
        .gettype        = segvn_gettype,
        .getvp          = segvn_getvp,
        .advise         = segvn_advise,
        .dump           = segvn_dump,
        .pagelock       = segvn_pagelock,
        .setpagesize    = segvn_setpagesize,
        .getmemid       = segvn_getmemid,
        .getpolicy      = segvn_getpolicy,
        .capable        = segvn_capable,
        .inherit        = segvn_inherit,
        segvn_dup,
        segvn_unmap,
        segvn_free,
        segvn_fault,
        segvn_faulta,
        segvn_setprot,
        segvn_checkprot,
        segvn_kluster,
        segvn_swapout,
        segvn_sync,
        segvn_incore,
        segvn_lockop,
        segvn_getprot,
        segvn_getoffset,
        segvn_gettype,
        segvn_getvp,
        segvn_advise,
        segvn_dump,
        segvn_pagelock,
        segvn_setpagesize,
        segvn_getmemid,
        segvn_getpolicy,
        segvn_capable,
        segvn_inherit
};
```
______***unchanged_portion_omitted_***

****
   16996 Tue Nov 24 09:34:39 2015
new/usr/src/uts/i86xpv/vm/seg_mf.c
6144 use C99 initializers in segment ops structures
****
______unchanged_portion_omitted_

```
static struct seg_ops segmf_ops = {
        .dup            = segmf_dup,
        .unmap          = segmf_unmap,
        .free           = segmf_free,
        .fault          = segmf_fault,
        .faulta         = segmf_faulta,
        .setprot        = segmf_setprot,
        .checkprot      = segmf_checkprot,
        .kluster        = segmf_kluster,
        .sync           = segmf_sync,
        .incore         = segmf_incore,
        .lockop         = segmf_lockop,
        .getprot        = segmf_getprot,
        .getoffset      = segmf_getoffset,
        .gettype        = segmf_gettype,
        .getvp          = segmf_getvp,
        .advise         = segmf_advise,
        .dump           = segmf_dump,
        .pagelock       = segmf_pagelock,
        .setpagesize    = segmf_setpagesize,
        .getmemid       = segmf_getmemid,
        .getpolicy      = segmf_getpolicy,
        .capable        = segmf_capable,
        .inherit        = seg_inherit_notsup,
        segmf_dup,
        segmf_unmap,
        segmf_free,
        segmf_fault,
        segmf_faulta,
        segmf_setprot,
        segmf_checkprot,
        (int (*)())segmf_kluster,
        (size_t (*)(struct seg *))NULL, /* swapout */
        segmf_sync,
        segmf_incore,
        segmf_lockop,
        segmf_getprot,
        segmf_getoffset,
        segmf_gettype,
        segmf_getvp,
        segmf_advise,
        segmf_dump,
        segmf_pagelock,
        segmf_setpagesize,
        segmf_getmemid,
        segmf_getpolicy,
        segmf_capable,
        seg_inherit_notsup
};
```

______unchanged_portion_omitted_

```
**********************************************************
   12471 Tue Nov 24 09:34:39 2015
new/usr/src/uts/sparc/v9/vm/seg_nf.c
6144 use C99 initializers in segment ops structures
**********************************************************
/*
 * 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.
 */

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

/*
 * 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,
        segnf_dup,
        segnf_unmap,
        segnf_free,
        (faultcode_t (*)(struct hat *, struct seg *, caddr_t, size_t,
            enum fault_type, enum seg_rw))
                segnf_nomap,            /* fault */
        (faultcode_t (*)(struct seg *, caddr_t))
                segnf_nomap,            /* faulta */
        segnf_setprot,
        segnf_checkprot,
        (int (*)())segnf_badop,         /* kluster */
        (size_t (*)(struct seg *))NULL, /* swapout */
        (int (*)(struct seg *, caddr_t, size_t, int, uint_t))
```

```
		segnf_nop,		/* sync */
	(size_t (*)(struct seg *, caddr_t, size_t, char *))
		segnf_nop,		/* incore */
	(int (*)(struct seg *, caddr_t, size_t, int, int, ulong_t *, size_t))
		segnf_nop,		/* lockop */
	segnf_getprot,
	segnf_getoffset,
	segnf_gettype,
	segnf_getvp,
	(int (*)(struct seg *, caddr_t, size_t, uint_t))
		segnf_nop,		/* advise */
	segnf_dump,
	segnf_pagelock,
	segnf_setpagesize,
	segnf_getmemid,
	segnf_getpolicy,
};
```
______**_unchanged_portion_omitted_**