```
**********************************************************
  116541 Mon Jan 14 13:17:03 2019
new/usr/src/uts/common/fs/vfs.c
10083 smatch fixes for common/fs/vfs.c
**********************************************************
/*
 * 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) 1988, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright (c) 2018, Joyent, Inc.
 * Copyright 2016 Joyent, Inc.
 * Copyright 2016 Toomas Soome <tsoome@me.com>
 * Copyright (c) 2016, 2017 by Delphix. All rights reserved.
 * Copyright 2016 Nexenta Systems, Inc.
 * Copyright 2017 RackTop Systems.
 */

/*      Copyright (c) 1983, 1984, 1985, 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.
 */

#include <sys/types.h>
#include <sys/t_lock.h>
#include <sys/param.h>
#include <sys/errno.h>
#include <sys/user.h>
#include <sys/fstyp.h>
#include <sys/kmem.h>
#include <sys/systm.h>
#include <sys/proc.h>
#include <sys/mount.h>
#include <sys/vfs.h>
#include <sys/vfs_opreg.h>
#include <sys/fem.h>
#include <sys/mntent.h>
#include <sys/stat.h>
#include <sys/statvfs.h>
#include <sys/statfs.h>
```

```
#include <sys/cred.h>
#include <sys/vnode.h>
#include <sys/rwstlock.h>
#include <sys/dnlc.h>
#include <sys/file.h>
#include <sys/time.h>
#include <sys/atomic.h>
#include <sys/cmn_err.h>
#include <sys/buf.h>
#include <sys/swap.h>
#include <sys/debug.h>
#include <sys/vnode.h>
#include <sys/modctl.h>
#include <sys/ddi.h>
#include <sys/pathname.h>
#include <sys/bootconf.h>
#include <sys/dumphdr.h>
#include <sys/dc_ki.h>
#include <sys/poll.h>
#include <sys/sunddi.h>
#include <sys/sysmacros.h>
#include <sys/zone.h>
#include <sys/policy.h>
#include <sys/ctfs.h>
#include <sys/objfs.h>
#include <sys/console.h>
#include <sys/reboot.h>
#include <sys/attr.h>
#include <sys/zio.h>
#include <sys/spa.h>
#include <sys/lofi.h>
#include <sys/bootprops.h>

#include <vm/page.h>

#include <fs/fs_subr.h>
/* Private interfaces to create vopstats-related data structures */
extern void             initialize_vopstats(vopstats_t *);
extern vopstats_t       *get_fstype_vopstats(struct vfs *, struct vfssw *);
extern vsk_anchor_t     *get_vskstat_anchor(struct vfs *);

static void vfs_clearmntopt_nolock(mntopts_t *, const char *, int);
static void vfs_setmntopt_nolock(mntopts_t *, const char *,
    const char *, int, int);
static int  vfs_optionisset_nolock(const mntopts_t *, const char *, char **);
static void vfs_freemnttab(struct vfs *);
static void vfs_freeopt(mntopt_t *);
static void vfs_swapopttbl_nolock(mntopts_t *, mntopts_t *);
static void vfs_swapopttbl(mntopts_t *, mntopts_t *);
static void vfs_copyopttbl_extend(const mntopts_t *, mntopts_t *, int);
static void vfs_createopttbl_extend(mntopts_t *, const char *,
    const mntopts_t *);
static char **vfs_copycancelopt_extend(char **const, int);
static void vfs_freecancelopt(char **);
static void getrootfs(char **, char **);
static int getmacpath(dev_info_t *, void *);
static void vfs_mnttabvp_setup(void);

struct ipmnt {
        struct ipmnt    *mip_next;
        dev_t           mip_dev;
        struct vfs      *mip_vfsp;
};
______unchanged_portion_omitted_

static void
```

```
lofi_remove(struct vfs *vfsp)
{
        struct lofi_ioctl *li = NULL;
        ldi_ident_t ldi_id;
        ldi_handle_t ldi_hdl;
        int err;

        if (vfsp->vfs_lofi_id == 0)
                return;

        ldi_id = ldi_ident_from_anon();

        li = kmem_zalloc(sizeof (*li), KM_SLEEP);
        li->li_id = vfsp->vfs_lofi_id;
        li->li_cleanup = B_TRUE;

        err = ldi_open_by_name("/dev/lofictl", FREAD | FWRITE, kcred,
            &ldi_hdl, ldi_id);

        if (err)
                goto out;

        err = ldi_ioctl(ldi_hdl, LOFI_UNMAP_FILE_MINOR, (intptr_t)li,
            FREAD | FWRITE | FKIOCTL, kcred, NULL);

        (void) ldi_close(ldi_hdl, FREAD | FWRITE, kcred);

        if (!err)
                vfsp->vfs_lofi_id = 0;

out:
        ldi_ident_release(ldi_id);
        if (li != NULL)
        kmem_free(li, sizeof (*li));
}
```

______***unchanged_portion_omitted_***

```
vfs_t EIO_vfs;
vfsops_t *EIO_vfsops;

/*
 * Called from startup() to initialize all loaded vfs's
 */
void
vfsinit(void)
{
        struct vfssw *vswp;
        int error;
        extern int vopstats_enabled;
        extern void vopstats_startup();

        static const fs_operation_def_t EIO_vfsops_template[] = {
                VFSNAME_MOUNT,          { .error = vfs_EIO },
                VFSNAME_UNMOUNT,        { .error = vfs_EIO },
                VFSNAME_ROOT,           { .error = vfs_EIO },
                VFSNAME_STATVFS,        { .error = vfs_EIO },
                VFSNAME_SYNC,           { .vfs_sync = vfs_EIO_sync },
                VFSNAME_VGET,           { .error = vfs_EIO },
                VFSNAME_MOUNTROOT,      { .error = vfs_EIO },
                VFSNAME_FREEVFS,        { .error = vfs_EIO },
                VFSNAME_VNSTATE,        { .error = vfs_EIO },
                NULL, NULL
        };

        static const fs_operation_def_t stray_vfsops_template[] = {
                VFSNAME_MOUNT,          { .error = vfsstray },
```

```
                VFSNAME_UNMOUNT,        { .error = vfsstray },
                VFSNAME_ROOT,           { .error = vfsstray },
                VFSNAME_STATVFS,        { .error = vfsstray },
                VFSNAME_SYNC,           { .vfs_sync = vfsstray_sync },
                VFSNAME_VGET,           { .error = vfsstray },
                VFSNAME_MOUNTROOT,      { .error = vfsstray },
                VFSNAME_FREEVFS,        { .error = vfsstray },
                VFSNAME_VNSTATE,        { .error = vfsstray },
                NULL, NULL
        };

        /* Create vfs cache */
        vfs_cache = kmem_cache_create("vfs_cache", sizeof (struct vfs),
            sizeof (uintptr_t), NULL, NULL, NULL, NULL, NULL, 0);

        /* Initialize the vnode cache (file systems may use it during init). */
        vn_create_cache();

        /* Setup event monitor framework */
        fem_init();

        /* Initialize the dummy stray file system type. */
        error = vfs_setfsops(0, stray_vfsops_template, NULL);

        /* Initialize the dummy EIO file system. */
        error = vfs_makefsops(EIO_vfsops_template, &EIO_vfsops);
        if (error != 0) {
                cmn_err(CE_WARN, "vfsinit: bad EIO vfs ops template");
                /* Shouldn't happen, but not bad enough to panic */
        }

        VFS_INIT(&EIO_vfs, EIO_vfsops, (caddr_t)NULL);

        /*
         * Default EIO_vfs.vfs_flag to VFS_UNMOUNTED so a lookup
         * on this vfs can immediately notice it's invalid.
         */
        EIO_vfs.vfs_flag |= VFS_UNMOUNTED;

        /*
         * Call the init routines of non-loadable filesystems only.
         * Filesystems which are loaded as separate modules will be
         * initialized by the module loading code instead.
         */

        for (vswp = &vfssw[1]; vswp < &vfssw[nfstype]; vswp++) {
                RLOCK_VFSSW();
                if (vswp->vsw_init != NULL)
                        (void) (*vswp->vsw_init)(vswp - vfssw, vswp->vsw_name);
                        (*vswp->vsw_init)(vswp - vfssw, vswp->vsw_name);
                RUNLOCK_VFSSW();
        }

        vopstats_startup();

        if (vopstats_enabled) {
                /* EIO_vfs can collect stats, but we don't retrieve them */
                initialize_vopstats(&EIO_vfs.vfs_vopstats);
                EIO_vfs.vfs_fstypevsp = NULL;
                EIO_vfs.vfs_vskap = NULL;
                EIO_vfs.vfs_flag |= VFS_STATS;
        }

        xattr_init();

        reparse_point_init();
```

```
4251 }
______unchanged_portion_omitted_
```