```
**********************************************************
   51356 Mon May  5 22:34:22 2014
new/usr/src/cmd/hal/hald/solaris/devinfo_storage.c
4846 HAL partition names don't match real partition names
**********************************************************
/***************************************************************************
 *
 * devinfo_storage.c : storage devices
 *
 * Copyright (c) 2006, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright 2013 Garrett D'Amore <garrett@damore.org>
 * Copyright 2014 Andrew Stormont.
#endif /* ! codereview */
 *
 * Licensed under the Academic Free License version 2.1
 *
 **************************************************************************/

#ifdef HAVE_CONFIG_H
#  include <config.h>
#endif

#include <stdio.h>
#include <string.h>
#include <strings.h>
#include <ctype.h>
#include <libdevinfo.h>
#include <sys/types.h>
#include <sys/mkdev.h>
#include <sys/stat.h>
#include <sys/mntent.h>
#include <sys/mnttab.h>

#include "../osspec.h"
#include "../logger.h"
#include "../hald.h"
#include "../hald_dbus.h"
#include "../device_info.h"
#include "../util.h"
#include "../hald_runner.h"
#include "hotplug.h"
#include "devinfo.h"
#include "devinfo_misc.h"
#include "devinfo_storage.h"
#include "osspec_solaris.h"

#ifdef sparc
#define WHOLE_DISK      "s2"
#else
#define WHOLE_DISK      "p0"
#endif

/* some devices,especially CDROMs, may take a while to be probed (values in ms)
#define DEVINFO_PROBE_STORAGE_TIMEOUT   60000
#define DEVINFO_PROBE_VOLUME_TIMEOUT    60000

typedef struct devinfo_storage_minor {
        char    *devpath;
        char    *devlink;
        char    *slice;
        dev_t   dev;
        int     dosnum; /* dos disk number or -1 */
} devinfo_storage_minor_t;

HalDevice *devinfo_ide_add(HalDevice *parent, di_node_t node, char *devfs_path,
static HalDevice *devinfo_ide_host_add(HalDevice *parent, di_node_t node, char *
```

```
static HalDevice *devinfo_ide_device_add(HalDevice *parent, di_node_t node, char
static HalDevice *devinfo_ide_storage_add(HalDevice *parent, di_node_t node, cha
HalDevice *devinfo_scsi_add(HalDevice *parent, di_node_t node, char *devfs_path,
static HalDevice *devinfo_scsi_storage_add(HalDevice *parent, di_node_t node, ch
HalDevice *devinfo_blkdev_add(HalDevice *parent, di_node_t node, char *devfs_pat
static HalDevice *devinfo_blkdev_storage_add(HalDevice *parent, di_node_t node,
HalDevice *devinfo_floppy_add(HalDevice *parent, di_node_t node, char *devfs_pat
static void devinfo_floppy_add_volume(HalDevice *parent, di_node_t node);
static HalDevice *devinfo_lofi_add(HalDevice *parent, di_node_t node, char *devf
static void devinfo_lofi_add_minor(HalDevice *parent, di_node_t node, char *mino
static void devinfo_storage_minors(HalDevice *parent, di_node_t node, gchar *dev
static struct devinfo_storage_minor *devinfo_storage_new_minor(char *maindev_pat
    char *devlink, dev_t dev, int dosnum);
static void devinfo_storage_free_minor(struct devinfo_storage_minor *m);
HalDevice *devinfo_volume_add(HalDevice *parent, di_node_t node, devinfo_storage
static void devinfo_volume_preprobing_done(HalDevice *d, gpointer userdata1, gpo
static void devinfo_volume_hotplug_begin_add (HalDevice *d, HalDevice *parent, D
static void devinfo_storage_hotplug_begin_add (HalDevice *d, HalDevice *parent,
static void devinfo_storage_probing_done (HalDevice *d, guint32 exit_type, gint
const gchar *devinfo_volume_get_prober (HalDevice *d, int *timeout);
const gchar *devinfo_storage_get_prober (HalDevice *d, int *timeout);

static char *devinfo_scsi_dtype2str(int dtype);
static char *devinfo_volume_get_slice_name (char *devlink);
static boolean_t is_dos_slice(const char *slice, int *partnum);
static gboolean dos_to_dev(char *path, char **devpath, int *partnum);
static gboolean is_dos_path(char *path, int *partnum);

static void devinfo_storage_set_nicknames (HalDevice *d);

DevinfoDevHandler devinfo_ide_handler = {
        devinfo_ide_add,
        NULL,
        NULL,
        NULL,
        NULL,
        NULL
};
______unchanged_portion_omitted_

/*
 * Storage minor nodes are potential "volume" objects.
 * This function also completes building the parent object (main storage device)
 */
static void
devinfo_storage_minors(HalDevice *parent, di_node_t node, gchar *devfs_path, gbo
{
        di_devlink_handle_t devlink_hdl;
        gboolean is_cdrom;
        const char *whole_disk;
        int     major;
        di_minor_t minor;
        dev_t   dev;
        char    *minor_path = NULL;
        char    *maindev_path = NULL;
        char    *devpath, *devlink;
        int     doslink_len;
        char    *doslink;
        char    dospath[64];
        char    *slice;
        int     pathlen;
        int     i;
        char    *raw;
        boolean_t maindev_is_d0;
        GQueue  *mq;
        HalDevice *volume;
        struct devinfo_storage_minor *m;
        struct devinfo_storage_minor *maindev = NULL;

        /* for cdroms whole disk is always s2 */
        is_cdrom = hal_device_has_capability (parent, "storage.cdrom");
        whole_disk = is_cdrom ? "s2" : WHOLE_DISK;

        major = di_driver_major(node);

        /* the "whole disk" p0/s2/d0 node must come first in the hotplug queue
         * so we put other minor nodes on the local queue and move to the
         * hotplug queue up in the end
         */
        if ((mq = g_queue_new()) == NULL) {
                goto err;
        }
        if ((devlink_hdl = di_devlink_init(NULL, 0)) == NULL) {
                g_queue_free (mq);
                goto err;
        }
        minor = DI_MINOR_NIL;
        while ((minor = di_minor_next(node, minor)) != DI_MINOR_NIL) {
                dev = di_minor_devt(minor);
                if ((major != major(dev)) ||
                    (di_minor_type(minor) != DDM_MINOR) ||
                    (di_minor_spectype(minor) != S_IFBLK) ||
                    ((minor_path = di_devfs_minor_path(minor)) == NULL)) {
                        continue;
                }
                if ((devlink = get_devlink(devlink_hdl, NULL, minor_path)) == NU
                        di_devfs_path_free (minor_path);
                        continue;
                }

                slice = devinfo_volume_get_slice_name (devlink);
                if (strlen (slice) < 2) {
                        free (devlink);
                        di_devfs_path_free (minor_path);
                        continue;
                }

                /* ignore p1..N - we'll use p0:N instead */
                if ((strlen (slice) > 1) && (slice[0] == 'p') && isdigit(slice[1
                    ((atol(&slice[1])) > 0)) {
                        free (devlink);
                        di_devfs_path_free (minor_path);
                        continue;
                }

                m = devinfo_storage_new_minor(minor_path, slice, devlink, dev, -
                if (m == NULL) {
                        free (devlink);
                        di_devfs_path_free (minor_path);
                        continue;
                }

                /* main device is either s2/p0 or d0, the latter taking preceden
                if ((strcmp (slice, "d0") == 0) ||
                    (((strcmp (slice, whole_disk) == 0) && (maindev == NULL)))) {
                        if (maindev_path != NULL) {
                                di_devfs_path_free (maindev_path);
                        }
                        maindev_path = minor_path;
                        maindev = m;
                        g_queue_push_head (mq, maindev);
                } else {
```

```
                        di_devfs_path_free (minor_path);
                        g_queue_push_tail (mq, m);
                }

                free (devlink);
        }
        di_devlink_fini (&devlink_hdl);

        if (maindev == NULL) {
                /* shouldn't typically happen */
                while (!g_queue_is_empty (mq)) {
                        devinfo_storage_free_minor (g_queue_pop_head (mq));
                }
                goto err;
        }

        /* first enqueue main storage device */
        if (!rescan) {
                hal_device_property_set_int (parent, "block.major", major);
                hal_device_property_set_int (parent, "block.minor", minor(mainde
                hal_device_property_set_string (parent, "block.device", maindev-
                raw = dsk_to_rdsk (maindev->devlink);
                hal_device_property_set_string (parent, "block.solaris.raw_devic
                free (raw);
                hal_device_property_set_bool (parent, "block.is_volume", FALSE);
                hal_device_property_set_string (parent, "solaris.devfs_path", ma
                devinfo_add_enqueue (parent, maindev_path, &devinfo_storage_hand
        }

        /* add virtual dos volumes to enable pcfs probing */
        if (!is_cdrom) {
                doslink_len = strlen (maindev->devlink) + sizeof (":NNN") + 1;
                if ((doslink = (char *)calloc (1, doslink_len)) != NULL) {
                        for (i = 1; i < 16; i++) {
                                snprintf(dospath, sizeof (dospath), "%s:%d", mai
                                snprintf(doslink, doslink_len, "%s:%d", maindev-
                                m = devinfo_storage_new_minor(maindev_path, dosp
                                g_queue_push_tail (mq, m);
                        }
                        free (doslink);
                }
        }

        maindev_is_d0 = (strcmp (maindev->slice, "d0") == 0);

        /* enqueue all volumes */
        while (!g_queue_is_empty (mq)) {
                m = g_queue_pop_head (mq);

                /* if main device is d0, we'll throw away s2/p0 */
                if (maindev_is_d0 && (strcmp (m->slice, whole_disk) == 0)) {
                        devinfo_storage_free_minor (m);
                        continue;
                }
                /* don't do p0 on cdrom */
                if (is_cdrom && (strcmp (m->slice, "p0") == 0)) {
                        devinfo_storage_free_minor (m);
                        continue;
                }
                if (rescan) {
                        /* in rescan mode, don't reprobe existing volumes */
                        /* XXX detect volume removal? */
                        volume = hal_device_store_match_key_value_string (hald_g
                            "solaris.devfs_path", m->devpath);
                        if ((volume == NULL) || !hal_device_has_capability(volum
                                devinfo_volume_add (parent, node, m);
                        } else {
                                HAL_INFO(("rescan volume exists %s", m->devpath)
                        }
                } else {
                        devinfo_volume_add (parent, node, m);
                }
                devinfo_storage_free_minor (m);
        }

        if (maindev_path != NULL) {
                di_devfs_path_free (maindev_path);
        }

        return;

err:
        if (maindev_path != NULL) {
                di_devfs_path_free (maindev_path);
        }
        if (!rescan) {
                devinfo_add_enqueue (parent, devfs_path, &devinfo_storage_handle
        }
}
______unchanged_portion_omitted_

static void
devinfo_volume_preprobing_done (HalDevice *d, gpointer userdata1, gpointer userd
{
        void *end_token = (void *) userdata1;
        char *whole_disk;
        char *block_device;
        const char *storage_udi;
        HalDevice *storage_d;
        const char *slice;
        int dos_num;

        if (hal_device_property_get_bool (d, "info.ignore")) {
                HAL_INFO (("Preprobing merged info.ignore==TRUE %s", hal_device_
                goto skip;
        }

        /*
         * Optimizations: only probe if there's a chance to find something
         */
        block_device = (char *)hal_device_property_get_string (d, "block.device"
        storage_udi = hal_device_property_get_string (d, "block.storage_device")
        slice = hal_device_property_get_string(d, "block.solaris.slice");
        if ((block_device == NULL) || (storage_udi == NULL) ||
            (slice == NULL) || (strlen (slice) < 2)) {
                HAL_INFO (("Malformed volume properties %s", hal_device_get_udi
                goto skip;
        }
        storage_d = hal_device_store_match_key_value_string (hald_get_gdl (), "i
        if (storage_d == NULL) {
                HAL_INFO (("Storage device not found %s", hal_device_get_udi (d)
                goto skip;
        }

        whole_disk = hal_device_has_capability (storage_d,
            "storage.cdrom") ? "s2" : WHOLE_DISK;

        if (is_dos_slice(slice, &dos_num)) {
        if (is_dos_path(block_device, &dos_num)) {
                /* don't probe more dos volumes than probe-storage found */
                if ((hal_device_property_get_bool (storage_d, "storage.no_partit
                    (dos_num > hal_device_property_get_int (storage_d, "storage.
```

```
                            HAL_INFO (("%d > %d %s", dos_num, hal_device_propert
                                "storage.solaris.num_dos_partitions"), hal_devic
                        goto skip;
                }
        } else {
                /* if no VTOC slices found, don't probe slices except s2 */
                if ((slice[0] == 's') && (isdigit(slice[1])) && ((strcmp (slice,
                    !hal_device_property_get_bool (storage_d, "storage.solaris.v
                        HAL_INFO (("Not probing slice %s", hal_device_get_udi (d
                        goto skip;
                }
        }

        HAL_INFO(("Probing udi=%s", hal_device_get_udi (d)));
        hald_runner_run (d,
                        "hald-probe-volume", NULL,
                        DEVINFO_PROBE_VOLUME_TIMEOUT,
                        devinfo_callouts_probing_done,
                        (gpointer) end_token, userdata2);

        return;

skip:
        hal_device_store_remove (hald_get_tdl (), d);
        g_object_unref (d);
        hotplug_event_end (end_token);
}
______unchanged_portion_omitted_

static boolean_t
is_dos_slice(const char *slice, int *partnum)
static gboolean
is_dos_path(char *path, int *partnum)
{
        char *p;

        if ((p = strrchr(slice, 'p')) == NULL &&
            (p = strrchr(slice, ':')) == NULL) {
                return (B_FALSE);
        if ((p = strrchr (path, ':')) == NULL) {
                return (FALSE);
        }

#endif /* ! codereview */
        return ((*partnum = atoi(p + 1)) != 0);
}

static gboolean
dos_to_dev(char *path, char **devpath, int *partnum)
{
        char *p;

        if ((p = strrchr (path, ':')) == NULL) {
                return (FALSE);
        }
        if ((*partnum = atoi(p + 1)) == 0) {
                return (FALSE);
        }
        p[0] = '\0';
        *devpath = strdup(path);
        p[0] = ':';
        return (*devpath != NULL);
}

static void
devinfo_storage_cleanup_mountpoint_cb (HalDevice *d, guint32 exit_type,
                       gint return_code, gchar **error,
                       gpointer data1, gpointer data2)
{
        char *mount_point = (char *) data1;

        HAL_INFO (("Cleaned up mount point ’%s’", mount_point));
        g_free (mount_point);
}
```
______***unchanged_portion_omitted***_

```
**********************************************************
    5151 Mon May  5 22:34:22 2014
new/usr/src/cmd/hal/utils/fsutils.c
4846 HAL partition names don't match real partition names
**********************************************************
/*
 *
 * fsutils.c : filesystem utilities
 *
 * Copyright 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 *
 * Copyright 2014 Andrew Stormont.
 *
#endif /* ! codereview */
 * Licensed under the Academic Free License version 2.1
 *
 */

#ifdef HAVE_CONFIG_H
#include <config.h>
#endif

#include <stdio.h>
#include <sys/types.h>
#include <sys/scsi/impl/uscsi.h>
#include <string.h>
#include <strings.h>
#include <ctype.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>
#include <fcntl.h>
#include <sys/dkio.h>
#include <libintl.h>
#include <sys/dktp/fdisk.h>
#include <sys/fs/pc_label.h>

#include <libhal.h>
#include "fsutils.h"

/*
 * Separates dos notation device spec into device and drive number
 *     pN partition names are rewritten to point to p0
 *     :N partition names are dropped
#endif /* ! codereview */
 */
boolean_t
dos_to_dev(char *path, char **devpath, int *num)
{
        int i;
        char *buf;
        boolean_t found = B_FALSE;
        char *p;

        for (i = strlen(path); i > 0; i--) {
                if (path[i] == 'p' || path[i] == ':') {
                        found = B_TRUE;
                        break;
                }
        if ((p = strrchr(path, ':')) == NULL) {
                return (B_FALSE);
        }

        if (found == B_FALSE || (*num = atoi(path + i + 1)) == 0 ||
            (buf = strdup(path)) == NULL) {
```

```
        if ((*num = atoi(p + 1)) == 0) {
                return (B_FALSE);
        }

        (void) strcpy(buf + i, path[i] == 'p' ? "p0" : "");
        *devpath = buf;
        return (B_TRUE);
        p[0] = '\0';
        *devpath = strdup(path);
        p[0] = ':';
        return (*devpath != NULL);
}
______unchanged_portion_omitted_
```