```
**********************************************************
   29189 Mon Feb 23 09:46:02 2015
new/usr/src/cmd/power/handlers.c
5526 One more gcc warning for cmd/power
**********************************************************
```

```
   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 2015 Gary Mills
  23  * Copyright 2009 Sun Microsystems, Inc.  All rights reserved.
  24  * Use is subject to license terms.
  25  */

  27 #include "pmconfig.h"
  28 #include <sys/mkdev.h>
  29 #include <sys/syslog.h>
  30 #include <sys/openpromio.h>
  31 #include <sys/mnttab.h>
  32 #include <sys/vtoc.h>
  33 #include <sys/efi_partition.h>
  34 #include <syslog.h>
  35 #include <stdlib.h>
  36 #include <sys/pm.h>
  37 #include <kstat.h>
  38 #include <sys/smbios.h>
  39 #include <libzfs.h>


  42 #define STRCPYLIM(dst, src, str) strcpy_limit(dst, src, sizeof (dst), str)
  43 #define LASTBYTE(str) (str + strlen(str) - 1)

  45 static char nerr_fmt[] = "number is out of range (%s)\n";
  46 static char alloc_fmt[] = "cannot allocate space for \"%s\", %s\n";
  47 static char set_thresh_fmt[] = "error setting threshold(s) for \"%s\", %s\n";
  48 static char bad_thresh_fmt[] = "bad threshold(s)\n";
  49 static char stat_fmt[] = "cannot stat \"%s\", %s\n";
  50 static char always_on[] = "always-on";

  52 #define PM_DEFAULT_ALGORITHM -1
  53 /*
  54  * When lines in a config file (usually "/etc/power.conf") start with
  55  * a recognized keyword, a "handler" routine is called for specific
  56  * CPR or PM -related action(s).  Each routine returns a status code
  57  * indicating whether all tasks were successful; if any errors occured,
  58  * future CPR or PM updates are skipped.  Following are the handler
  59  * routines for all keywords:
  60  */
```



```
  63 static char pm_cmd_string[32];

  65 static char *
  66 pm_map(int cmd)
  67 {
  68         pm_req_t req;

  70         req.value = cmd;
  71         req.data = (void *)pm_cmd_string;
  72         req.datasize = sizeof (pm_cmd_string);

  74         if (ioctl(pm_fd, PM_GET_CMD_NAME, &req) < 0) {
  75                 perror(gettext("PM_GET_CMD_NAME failed:"));
  76                 return ("??");
  77         }
  78         return (pm_cmd_string);
  79 }
_____unchanged_portion_omitted_

 998 /*
 999  * given the path to a zvol, return the cXtYdZ name
1000  * returns < 0 on error, 0 if it isn't a zvol, > 1 on success
1001  */
1002 static int
1003 ztop(char *arg, char *diskname)
1004 {
1005         zpool_handle_t *zpool_handle;
1006         nvlist_t *config, *nvroot;
1007         nvlist_t **child;
1008         uint_t children;
1009         libzfs_handle_t *lzfs;
1010         char *vname;
1011         char *p;
1012         char pool_name[MAXPATHLEN];

1014         if (strncmp(arg, "/dev/zvol/dsk/", 14)) {
1015                 return (0);
1016         }
1017         arg += 14;
1018         (void) strncpy(pool_name, arg, MAXPATHLEN);
1019         if ((p = strchr(pool_name, '/')) != NULL)
1018         if (p = strchr(pool_name, '/'))
1020                 *p = '\0';
1021         STRCPYLIM(new_cc.cf_fs, p + 1, "statefile path");

1023         if ((lzfs = libzfs_init()) == NULL) {
1024                 mesg(MERR, "failed to initialize ZFS library\n");
1025                 return (-1);
1026         }
1027         if ((zpool_handle = zpool_open(lzfs, pool_name)) == NULL) {
1028                 mesg(MERR, "couldn't open pool '%s'\n", pool_name);
1029                 libzfs_fini(lzfs);
1030                 return (-1);
1031         }
1032         config = zpool_get_config(zpool_handle, NULL);
1033         if (nvlist_lookup_nvlist(config, ZPOOL_CONFIG_VDEV_TREE,
1034             &nvroot) != 0) {
1035                 zpool_close(zpool_handle);
1036                 libzfs_fini(lzfs);
1037                 return (-1);
1038         }
1039         verify(nvlist_lookup_nvlist_array(nvroot, ZPOOL_CONFIG_CHILDREN,
1040             &child, &children) == 0);
1041         if (children != 1) {
1042                 mesg(MERR, "expected one vdev, got %d\n", children);
```

```
1043                 zpool_close(zpool_handle);
1044                 libzfs_fini(lzfs);
1045                 return (-1);
1046         }
1047         vname = zpool_vdev_name(lzfs, zpool_handle, child[0], B_FALSE);
1048         if (vname == NULL) {
1049                 mesg(MERR, "couldn't determine vdev name\n");
1050                 zpool_close(zpool_handle);
1051                 libzfs_fini(lzfs);
1052                 return (-1);
1053         }
1054         (void) strcpy(diskname, "/dev/dsk/");
1055         (void) strcat(diskname, vname);
1056         free(vname);
1057         zpool_close(zpool_handle);
1058         libzfs_fini(lzfs);
1059         return (1);
1060 }
```

_____***unchanged_portion_omitted_***