```
**********************************************************
    6129 Mon Aug 10 20:40:28 2015
new/usr/src/cmd/dumpadm/main.c
6110 dumpadm usage string should mention '-d none'
**********************************************************
```

```
/*
 * 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) 1998, 2010, Oracle and/or its affiliates. All rights reserved.
 * Copyright 2015 Nexenta Systems, Inc. All rights reserved.
 */

#include <sys/stat.h>
#include <locale.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include "dconf.h"
#include "minfree.h"
#include "utils.h"

static const char USAGE[] = "\
Usage: %s [-enuy] [-c kernel | curproc | all ]\n\
        [-d dump-device | swap | none ] [-m min {k|m|%%} ] [-s savecore-dir]\n\
        [-r root-dir] [-z on|off]\n";
Usage: %s [-enuy] [-c kernel | curproc | all ] [-d dump-device | swap ]\n\
        [-m min {k|m|%%} ] [-s savecore-dir] [-r root-dir] [-z on|off]\n";

static const char OPTS[] = "einuyc:d:m:s:r:z:";

static const char PATH_DEVICE[] = "/dev/dump";
static const char PATH_CONFIG[] = "/etc/dumpadm.conf";

int
main(int argc, char *argv[])
{
        const char *pname = getpname(argv[0]);

        u_longlong_t minf;
        struct stat st;
        int c;
        int dflag = 0;                  /* for checking in use during -d ops */
        int eflag = 0;                  /* print estimated dump size */
        int dcmode = DC_CURRENT;        /* kernel settings override unless -u */
        int modified = 0;               /* have we modified the dump config? */
        char *minfstr = NULL;           /* string value of -m argument */
```



```
        dumpconf_t dc;                  /* current configuration */
        int chrooted = 0;
        int douuid = 0;

        (void) setlocale(LC_ALL, "");
        (void) textdomain(TEXT_DOMAIN);

        /*
         * Take an initial lap through argv hunting for -r root-dir,
         * so that we can chroot before opening the configuration file.
         * We also handle -u and any bad options at this point.
         */
        while (optind < argc) {
                while ((c = getopt(argc, argv, OPTS)) != (int)EOF) {
                        if (c == 'r' && ++chrooted && chroot(optarg) == -1)
                                die(gettext("failed to chroot to %s"), optarg);
                        else if (c == 'u')
                                dcmode = DC_OVERRIDE;
                        else if (c == '?') {
                                (void) fprintf(stderr, gettext(USAGE), pname);
                                return (E_USAGE);
                        }
                }

                if (optind < argc) {
                        warn(gettext("illegal argument -- %s\n"), argv[optind]);
                        (void) fprintf(stderr, gettext(USAGE), pname);
                        return (E_USAGE);
                }
        }

        if (geteuid() != 0)
                die(gettext("you must be root to use %s\n"), pname);

        /*
         * If no config file exists yet, we're going to create an empty one,
         * so set the modified flag to force writing out the file.
         */
        if (access(PATH_CONFIG, F_OK) == -1)
                modified++;

        /*
         * Now open and read in the initial values from the config file.
         * If it doesn't exist, we create an empty file and dc is
         * initialized with the default values.
         */
        if (dconf_open(&dc, PATH_DEVICE, PATH_CONFIG, dcmode) == -1)
                return (E_ERROR);

        /*
         * Take another lap through argv, processing options and
         * modifying the dumpconf_t as appropriate.
         */
        for (optind = 1; optind < argc; optind++) {
                while ((c = getopt(argc, argv, OPTS)) != (int)EOF) {
                        switch (c) {
                        case 'c':
                                if (dconf_str2content(&dc, optarg) == -1)
                                        return (E_USAGE);
                                modified++;
                                break;
                        case 'd':
                                if (dconf_str2device(&dc, optarg) == -1)
                                        return (E_USAGE);
                                dflag++;
                                modified++;
```

```
126                             break;
127                     case 'e':
128                             eflag++;
129                             break;
130                     case 'i':
131                             /* undocumented option */
132                             if (chrooted) {
133                                     warn(gettext("-i and -r cannot be "
134                                         "used together\n"));
135                                     return (E_USAGE);
136                             }
137                             douuid++;
138                             break;
139
140                     case 'm':
141                             minfstr = optarg;
142                             break;
143
144                     case 'n':
145                             dc.dc_enable = DC_OFF;
146                             modified++;
147                             break;
148
149                     case 's':
150                             if (stat(optarg, &st) == -1 ||
151                                 !S_ISDIR(st.st_mode)) {
152                                     warn(gettext("%s is missing or not a "
153                                         "directory\n"), optarg);
154                                     return (E_USAGE);
155                             }
156
157                             if (dconf_str2savdir(&dc, optarg) == -1)
158                                     return (E_USAGE);
159                             modified++;
160                             break;
161
162                     case 'y':
163                             dc.dc_enable = DC_ON;
164                             modified++;
165                             break;
166
167                     case 'z':
168                             if (dconf_str2csave(&dc, optarg) == -1)
169                                     return (E_USAGE);
170                             modified++;
171                             break;
172                     }
173             }
174     }
175
176     if (eflag) {
177             if (argc == 2 && argv[1][0] == '-' && argv[1][1] == 'e' &&
178                 !argv[1][2])
179                     return (dconf_get_dumpsize(&dc) ? E_SUCCESS : E_ERROR);
180             else
181                     die(gettext("-e cannot be used with other options\n"));
182     }
183
184     if (douuid)
185             return (dconf_write_uuid(&dc) ? E_SUCCESS : E_ERROR);
186
187     if (minfstr != NULL) {
188             if (minfree_compute(dc.dc_savdir, minfstr, &minf) == -1)
189                     return (E_USAGE);
190             if (minfree_write(dc.dc_savdir, minf) == -1)
191                     return (E_ERROR);
```



```
192     }
193
194     if (dcmode == DC_OVERRIDE) {
195             /*
196              * In override mode, we try to force an update.  If this
197              * fails, we re-load the kernel configuration and write that
198              * out to the file in order to force the file in sync.
199              *
200              * We allow the file to be read-only but print a warning to the
201              * user that indicates it hasn't been updated.
202              */
203             if (dconf_update(&dc, 0) == -1)
204                     (void) dconf_getdev(&dc);
205             if (dc.dc_readonly)
206                     warn(gettext("kernel settings updated, but "
207                         "%s is read-only\n"), PATH_CONFIG);
208             else if (dconf_write(&dc) == -1)
209                     return (E_ERROR);
210
211     } else if (modified) {
212             /*
213              * If we're modifying the configuration, then try
214              * to update it, and write out the file if successful.
215              */
216             if (dc.dc_readonly) {
217                     warn(gettext("failed to update settings: %s is "
218                         "read-only\n"), PATH_CONFIG);
219                     return (E_ERROR);
220             }
221
222             if (dconf_update(&dc, dflag) == -1 ||
223                 dconf_write(&dc) == -1)
224                     return (E_ERROR);
225     }
226
227     if (dcmode == DC_CURRENT)
228             dconf_print(&dc, stdout);
229
230     if (dconf_close(&dc) == -1)
231             warn(gettext("failed to close configuration file"));
232
233     return (E_SUCCESS);
234 }
```

_____***unchanged_portion_omitted_***