```
**********************************************************
    8177 Thu Jan 17 15:03:52 2019
new/usr/src/lib/libcmdutils/common/process_xattrs.c
10151 mv_xattrs() checks for a NULL array name
**********************************************************
```

```
/*
 * 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 2008 Sun Microsystems, Inc.  All rights reserved.
 * Use is subject to license terms.
 * Copyright 2012 Milan Jurik. All rights reserved.
 * Copyright (c) 2018, Joyent, Inc.
 */

#include "libcmdutils.h"


/*
 * Gets file descriptors of attribute directories for source and target
 * attribute files
 */
int
get_attrdirs(int indfd, int outdfd, char *attrfile, int *sfd, int *tfd)
{
        int     pwdfd;
        int     fd1;
        int     fd2;

        pwdfd = open(".", O_RDONLY);
        if ((pwdfd != -1) && (fchdir(indfd) == 0)) {
                if ((fd1 = attropen(attrfile, ".", O_RDONLY)) == -1) {
                        (void) fchdir(pwdfd);
                        (void) close(pwdfd);
                        return (1);
                }
                *sfd = fd1;
        } else {
                (void) fchdir(pwdfd);
                (void) close(pwdfd);
                return (1);
        }
        if (fchdir(outdfd) == 0) {
                if ((fd2 = attropen(attrfile, ".", O_RDONLY)) == -1) {
                        (void) fchdir(pwdfd);
                        (void) close(pwdfd);
                        return (1);
                }
```



```
                *tfd = fd2;
        } else {
                (void) fchdir(pwdfd);
                (void) close(pwdfd);
                return (1);
        }
        (void) fchdir(pwdfd);
        return (0);
}

/*
 * mv_xattrs - Copies the content of the extended attribute files. Then
 *      moves the extended system attributes from the input attribute files
 *      to the target attribute files. Moves the extended system attributes
 *      from source to the target file. This function returns 0 on success
 *      and nonzero on error.
 */
int
mv_xattrs(char *cmd, char *infile, char *outfile, int sattr, int silent)
{
        int srcfd = -1;
        int indfd = -1;
        int outdfd = -1;
        int tmpfd = -1;
        int sattrfd = -1;
        int tattrfd = -1;
        int asfd = -1;
        int atfd = -1;
        DIR *dirp = NULL;
        struct dirent *dp = NULL;
        char *etext = NULL;
        struct stat st1;
        struct stat st2;
        nvlist_t *response = NULL;
        nvlist_t *res = NULL;

        if ((srcfd = open(infile, O_RDONLY)) == -1) {
                etext = dgettext(TEXT_DOMAIN, "cannot open source");
                goto error;
        }
        if (sattr)
                response = sysattr_list(cmd, srcfd, infile);

        if ((indfd = openat(srcfd, ".", O_RDONLY|O_XATTR)) == -1) {
                etext = dgettext(TEXT_DOMAIN, "cannot openat source");
                goto error;
        }
        if ((outdfd = attropen(outfile, ".", O_RDONLY)) == -1) {
                etext = dgettext(TEXT_DOMAIN, "cannot attropen target");
                goto error;
        }
        if ((tmpfd = dup(indfd)) == -1) {
                etext = dgettext(TEXT_DOMAIN, "cannot dup descriptor");
                goto error;

        }
        if ((dirp = fdopendir(tmpfd)) == NULL) {
                etext = dgettext(TEXT_DOMAIN, "cannot access source");
                goto error;
        }
        while ((dp = readdir(dirp)) != NULL) {
                if ((dp->d_name[0] == '.' && dp->d_name[1] == '\0') ||
                    (dp->d_name[0] == '.' && dp->d_name[1] == '.' &&
                    dp->d_name[2] == '\0') ||
                    (sysattr_type(dp->d_name) == _RO_SATTR) ||
                    (sysattr_type(dp->d_name) == _RW_SATTR))
```

```
128                         continue;

130                 if ((sattrfd = openat(indfd, dp->d_name,
131                     O_RDONLY)) == -1) {
132                         etext = dgettext(TEXT_DOMAIN,
133                             "cannot open src attribute file");
134                         goto error;
135                 }
136                 if (fstat(sattrfd, &st1) < 0) {
137                         etext = dgettext(TEXT_DOMAIN,
138                             "could not stat attribute file");
139                         goto error;
140                 }
141                 if ((tattrfd = openat(outdfd, dp->d_name,
142                     O_RDWR|O_CREAT|O_TRUNC, st1.st_mode)) == -1) {
143                         etext = dgettext(TEXT_DOMAIN,
144                             "cannot open target attribute file");
145                         goto error;
146                 }
147                 if (fstat(tattrfd, &st2) < 0) {
148                         etext = dgettext(TEXT_DOMAIN,
149                             "could not stat attribute file");
150                         goto error;
151                 }
152                 if (writefile(sattrfd, tattrfd, infile, outfile, dp->d_name,
153                     dp->d_name, &st1, &st2) != 0) {
154                         etext = dgettext(TEXT_DOMAIN,
155                             "failed to copy extended attribute "
156                             "from source to target");
157                         goto error;
158                 }

160                 errno = 0;
161                 if (sattr) {
162                         /*
163                          * Gets non default extended system attributes from
164                          * source to copy to target.
165                          */
165                         if (dp->d_name != NULL)
166                         res = sysattr_list(cmd, sattrfd, dp->d_name);

168                         if (res != NULL &&
169                             get_attrdirs(indfd, outdfd, dp->d_name, &asfd,
170                             &atfd) != 0) {
171                                 etext = dgettext(TEXT_DOMAIN,
172                                     "Failed to open attribute files");
173                                 goto error;
174                         }
175                         /*
176                          * Copy extended system attribute from source
177                          * attribute file to target attribute file
178                          */
179                         if (res != NULL &&
180                             (renameat(asfd, VIEW_READWRITE, atfd,
181                             VIEW_READWRITE) != 0)) {
182                                 if (errno == EPERM)
183                                         etext = dgettext(TEXT_DOMAIN,
184                                             "Permission denied -"
185                                             "failed to move system attribute");
186                                 else
187                                         etext = dgettext(TEXT_DOMAIN,
188                                             "failed to move extended "
189                                             "system attribute");
190                                 goto error;
191                         }
192                 }
```



```
193                 if (sattrfd != -1)
194                         (void) close(sattrfd);
195                 if (tattrfd != -1)
196                         (void) close(tattrfd);
197                 if (asfd != -1)
198                         (void) close(asfd);
199                 if (atfd != -1)
200                         (void) close(atfd);
201                 if (res != NULL) {
202                         nvlist_free(res);
203                         res = NULL;
204                 }
205         }
206         errno = 0;
207         /* Copy extended system attribute from source to target */

209         if (response != NULL) {
210                 if (renameat(indfd, VIEW_READWRITE, outdfd,
211                     VIEW_READWRITE) == 0)
212                         goto done;

214                 if (errno == EPERM)
215                         etext = dgettext(TEXT_DOMAIN, "Permission denied");
216                 else
217                         etext = dgettext(TEXT_DOMAIN,
218                             "failed to move system attribute");
219         }
220 error:
221         nvlist_free(res);
222         if (silent == 0 && etext != NULL) {
223                 if (!sattr)
224                         (void) fprintf(stderr, dgettext(TEXT_DOMAIN,
225                             "%s: %s: cannot move extended attributes, "),
226                             cmd, infile);
227                 else
228                         (void) fprintf(stderr, dgettext(TEXT_DOMAIN,
229                             "%s: %s: cannot move extended system "
230                             "attributes, "), cmd, infile);
231                 perror(etext);
232         }
233 done:
234         if (dirp)
235                 (void) closedir(dirp);
236         if (sattrfd != -1)
237                 (void) close(sattrfd);
238         if (tattrfd != -1)
239                 (void) close(tattrfd);
240         if (asfd != -1)
241                 (void) close(asfd);
242         if (atfd != -1)
243                 (void) close(atfd);
244         if (indfd != -1)
245                 (void) close(indfd);
246         if (outdfd != -1)
247                 (void) close(outdfd);
248         nvlist_free(response);
249         if (etext != NULL)
250                 return (1);
251         else
252                 return (0);
253 }
```

_____***unchanged_portion_omitted_***