```
**********************************************************
    5969 Sun Mar  8 10:35:15 2015
new/usr/src/lib/libcmdutils/common/writefile.c
1150 libcmdutils has superfluous #define
Reviewed by: Josef 'Jeff' Sipek <josef.sipek@nexenta.com>
Reviewed by: Andy Stormont <astormont@racktopsystems.com>
Reviewed by: Marcel Telka <marcel@telka.sk>
**********************************************************
```

```
/*
 * 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 (c) 1984, 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.
 */

#pragma ident   "%Z%%M% %I%     %E% SMI"

#include "libcmdutils.h"


int
writefile(int fi, int fo, char *infile, char *outfile, char *asfile,
    char *atfile, struct stat *s1p, struct stat *s2p)
{
        int mapsize, munmapsize;
        caddr_t cp;
        off_t filesize = s1p->st_size;
        off_t offset;
        int nbytes;
        int remains;
        int n;
        size_t src_size;
        size_t targ_size;
        char *srcbuf;
```



```
        char *targbuf;

        if (asfile != NULL) {
                src_size = strlen(infile) + strlen(asfile) +
                    strlen(dgettext(TEXT_DOMAIN, " attribute ")) + 1;
        } else {
                src_size = strlen(infile) + 1;
        }
        srcbuf = malloc(src_size);
        if (srcbuf == NULL) {
                (void) fprintf(stderr,
                    dgettext(TEXT_DOMAIN, "could not allocate memory"
                    " for path buffer: "));
                return (1);
        }
        if (asfile != NULL) {
                (void) snprintf(srcbuf, src_size, "%s%s%s",
                    infile, dgettext(TEXT_DOMAIN, " attribute "), asfile);
        } else {
                (void) snprintf(srcbuf, src_size, "%s", infile);
        }

        if (atfile != NULL) {
                targ_size = strlen(outfile) + strlen(atfile) +
                    strlen(dgettext(TEXT_DOMAIN, " attribute ")) + 1;
        } else {
                targ_size = strlen(outfile) + 1;
        }
        targbuf = malloc(targ_size);
        if (targbuf == NULL) {
                (void) fprintf(stderr,
                    dgettext(TEXT_DOMAIN, "could not allocate memory"
                    " for path buffer: "));
                return (1);
        }
        if (atfile != NULL) {
                (void) snprintf(targbuf, targ_size, "%s%s%s",
                    outfile, dgettext(TEXT_DOMAIN, " attribute "), atfile);
        } else {
                (void) snprintf(targbuf, targ_size, "%s", outfile);
        }

        if (S_ISREG(s1p->st_mode) && s1p->st_size > SMALLFILESIZE) {
        if (ISREG(*s1p) && s1p->st_size > SMALLFILESIZE) {
                /*
                 * Determine size of initial mapping.  This will determine the
                 * size of the address space chunk we work with.  This initial
                 * mapping size will be used to perform munmap() in the future.
                 */
                mapsize = MAXMAPSIZE;
                if (s1p->st_size < mapsize) mapsize = s1p->st_size;
                munmapsize = mapsize;

                /*
                 * Mmap time!
                 */
                if ((cp = mmap((caddr_t)NULL, mapsize, PROT_READ,
                    MAP_SHARED, fi, (off_t)0)) == MAP_FAILED)
                        mapsize = 0;   /* can't mmap today */
        } else
                mapsize = 0;

        if (mapsize != 0) {
                offset = 0;

                for (;;) {
```

```
122                         nbytes = write(fo, cp, mapsize);
123                         /*
124                          * if we write less than the mmaped size it's due to a
125                          * media error on the input file or out of space on
126                          * the output file.  So, try again, and look for errno.
127                          */
128                         if ((nbytes >= 0) && (nbytes != (int)mapsize)) {
129                                 remains = mapsize - nbytes;
130                                 while (remains > 0) {
131                                         nbytes = write(fo,
132                                             cp + mapsize - remains, remains);
133                                         if (nbytes < 0) {
134                                                 if (errno == ENOSPC)
135                                                         perror(targbuf);
136                                                 else
137                                                         perror(srcbuf);
138                                                 (void) close(fi);
139                                                 (void) close(fo);
140                                                 (void) munmap(cp, munmapsize);
141                                                 if (S_ISREG(s2p->st_mode))
143                                                 if (ISREG(*s2p))
142                                                         (void) unlink(targbuf);
143                                                 return (1);
144                                         }
145                                         remains -= nbytes;
146                                         if (remains == 0)
147                                                 nbytes = mapsize;
148                                 }
149                         }
150                         /*
151                          * although the write manual page doesn't specify this
152                          * as a possible errno, it is set when the nfs read
153                          * via the mmap'ed file is accessed, so report the
154                          * problem as a source access problem, not a target file
155                          * problem
156                          */
157                         if (nbytes < 0) {
158                                 if (errno == EACCES)
159                                         perror(srcbuf);
160                                 else
161                                         perror(targbuf);
162                                 (void) close(fi);
163                                 (void) close(fo);
164                                 (void) munmap(cp, munmapsize);
165                                 if (S_ISREG(s2p->st_mode))
167                                 if (ISREG(*s2p))
166                                         (void) unlink(targbuf);
167                                 if (srcbuf != NULL)
168                                         free(srcbuf);
169                                 if (targbuf != NULL)
170                                         free(targbuf);
171                                 return (1);
172                         }
173                         filesize -= nbytes;
174                         if (filesize == 0)
175                                 break;
176                         offset += nbytes;
177                         if (filesize < mapsize)
178                                 mapsize = filesize;
179                         if (mmap(cp, mapsize, PROT_READ, MAP_SHARED |
180                             MAP_FIXED, fi, offset) == MAP_FAILED) {
181                                 perror(srcbuf);
182                                 (void) close(fi);
183                                 (void) close(fo);
184                                 (void) munmap(cp, munmapsize);
185                                 if (S_ISREG(s2p->st_mode))
```



```
187                                 if (ISREG(*s2p))
186                                         (void) unlink(targbuf);
187                                 if (srcbuf != NULL)
188                                         free(srcbuf);
189                                 if (targbuf != NULL)
190                                         free(targbuf);
191                                 return (1);
192                         }
193                 }
194                 (void) munmap(cp, munmapsize);
195         } else {
196                 char buf[SMALLFILESIZE];
197                 for (;;) {
198                         n = read(fi, buf, sizeof (buf));
199                         if (n == 0) {
200                                 return (0);
201                         } else if (n < 0) {
202                                 (void) close(fi);
203                                 (void) close(fo);
204                                 if (S_ISREG(s2p->st_mode))
206                                 if (ISREG(*s2p))
205                                         (void) unlink(targbuf);
206                                 if (srcbuf != NULL)
207                                         free(srcbuf);
208                                 if (targbuf != NULL)
209                                         free(targbuf);
210                                 return (1);
211                         } else if (write(fo, buf, n) != n) {
212                                 (void) close(fi);
213                                 (void) close(fo);
214                                 if (S_ISREG(s2p->st_mode))
216                                 if (ISREG(*s2p))
215                                         (void) unlink(targbuf);
216                                 if (srcbuf != NULL)
217                                         free(srcbuf);
218                                 if (targbuf != NULL)
219                                         free(targbuf);
220                                 return (1);
221                         }
222                 }
223         }
224         if (srcbuf != NULL)
225                 free(srcbuf);
226         if (targbuf != NULL)
227                 free(targbuf);
228         return (0);
229 }
```

_____unchanged_portion_omitted_

```
**********************************************************
    4951 Sun Mar  8 10:35:16 2015
new/usr/src/lib/libcmdutils/libcmdutils.h
1150 libcmdutils has superfluous #define
Reviewed by: Josef 'Jeff' Sipek <josef.sipek@nexenta.com>
Reviewed by: Andy Stormont <astormont@racktopsystems.com>
Reviewed by: Marcel Telka <marcel@telka.sk>
**********************************************************
```

```
   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 2007 Sun Microsystems, Inc.  All rights reserved.
  23  * Use is subject to license terms.
  24  */
  25 /*
  26  * Copyright (c) 2013 RackTop Systems.
  27  */
  28 /*
  29  * Copyright 2014 Joyent, Inc.
  30  */

  32 /*
  33  * Declarations for the functions in libcmdutils.
  34  */

  36 #ifndef _LIBCMDUTILS_H
  37 #define _LIBCMDUTILS_H

  39 /*
  40  * This is a private header file.  Applications should not directly include
  41  * this file.
  42  */

  44 #include <stdio.h>
  45 #include <unistd.h>
  46 #include <stdlib.h>
  47 #include <errno.h>
  48 #include <fcntl.h>
  49 #include <limits.h>
  50 #include <libintl.h>
  51 #include <string.h>
  52 #include <dirent.h>
  53 #include <attr.h>
  54 #include <sys/avl.h>
  55 #include <sys/types.h>
  56 #include <sys/stat.h>
  57 #include <sys/mman.h>
  58 #include <libnvpair.h>
```



```
  60 #ifdef  __cplusplus
  61 extern "C" {
  62 #endif

  64 /* extended system attribute support */
  65 #define _NOT_SATTR      0
  66 #define _RO_SATTR       1
  67 #define _RW_SATTR       2

  69 #define MAXMAPSIZE      (1024*1024*8)   /* map at most 8MB */
  70 #define SMALLFILESIZE   (32*1024)       /* don't use mmap on little file */
  71 #define ISREG(A)        (((A).st_mode & S_IFMT) == S_IFREG)

  72 /* avltree */
  73 #define OFFSETOF(s, m)  ((size_t)(&(((s *)0)->m)))

  75 /* Type used for a node containing a device id and inode number */
  76 typedef struct tree_node {
  77         dev_t           node_dev;
  78         ino_t           node_ino;
  79         avl_node_t      avl_link;
  80 } tree_node_t;
______unchanged_portion_omitted_
```