

```

*****
5969 Sun Mar 1 12:47:15 2015
new/usr/src/lib/libcndutils/common/writefile.c
1150 libcndutils has superfluous #define
*****
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 */

22 /*
23  * Copyright 2008 Sun Microsystems, Inc. All rights reserved.
24  * Use is subject to license terms.
25  */

27 /*      Copyright (c) 1984, 1986, 1987, 1988, 1989 AT&T */
28 /*      All Rights Reserved */

30 /*
31  * University Copyright- Copyright (c) 1982, 1986, 1988
32  * The Regents of the University of California
33  * All Rights Reserved
34  *
35  * University Acknowledgment- Portions of this document are derived from
36  * software developed by the University of California, Berkeley, and its
37  * contributors.
38  */

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

40 #include "libcndutils.h"

43 int
44 writefile(int fi, int fo, char *infile, char *outfile, char *asfile,
45          char *atfile, struct stat *slp, struct stat *s2p)
46 {
47     int mapsize, munmapsize;
48     caddr_t cp;
49     off_t filesize = slp->st_size;
50     off_t offset;
51     int nbytes;
52     int remains;
53     int n;
54     size_t src_size;
55     size_t targ_size;
56     char *srcbuf;
57     char *targbuf;

59     if (asfile != NULL) {

```

```

60         src_size = strlen(infile) + strlen(asfile) +
61         strlen(dgettext(TEXT_DOMAIN, " attribute ")) + 1;
62     } else {
63         src_size = strlen(infile) + 1;
64     }
65     srcbuf = malloc(src_size);
66     if (srcbuf == NULL) {
67         (void) fprintf(stderr,
68             dgettext(TEXT_DOMAIN, "could not allocate memory"
69                 " for path buffer: "));
70         return (1);
71     }
72     if (asfile != NULL) {
73         (void) snprintf(srcbuf, src_size, "%s%s",
74             infile, dgettext(TEXT_DOMAIN, " attribute "), asfile);
75     } else {
76         (void) snprintf(srcbuf, src_size, "%s", infile);
77     }

79     if (atfile != NULL) {
80         targ_size = strlen(outfile) + strlen(atfile) +
81         strlen(dgettext(TEXT_DOMAIN, " attribute ")) + 1;
82     } else {
83         targ_size = strlen(outfile) + 1;
84     }
85     targbuf = malloc(targ_size);
86     if (targbuf == NULL) {
87         (void) fprintf(stderr,
88             dgettext(TEXT_DOMAIN, "could not allocate memory"
89                 " for path buffer: "));
90         return (1);
91     }
92     if (atfile != NULL) {
93         (void) snprintf(targbuf, targ_size, "%s%s",
94             outfile, dgettext(TEXT_DOMAIN, " attribute "), atfile);
95     } else {
96         (void) snprintf(targbuf, targ_size, "%s", outfile);
97     }

99     if (S_ISREG(slp->st_mode) && slp->st_size > SMALLFILESIZE) {
101     if (ISREG(*slp) && slp->st_size > SMALLFILESIZE) {
102         /*
103          * Determine size of initial mapping. This will determine the
104          * size of the address space chunk we work with. This initial
105          * mapping size will be used to perform munmap() in the future.
106          */
107         mapsize = MAXMAPSIZE;
108         if (slp->st_size < mapsize) mapsize = slp->st_size;
109         munmapsize = mapsize;

109         /*
110          * Mmap time!
111          */
112         if ((cp = mmap((caddr_t)NULL, mapsize, PROT_READ,
113             MAP_SHARED, fi, (off_t)0)) == MAP_FAILED)
114             mapsize = 0; /* can't mmap today */
115     } else
116         mapsize = 0;

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

121         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))
142                     if (ISREG(*s2p))
143                         (void) unlink(targbuf);
144                 return (1);
145             }
146             remains -= nbytes;
147             if (remains == 0)
148                 nbytes = mapsize;
149         }
150     }
151     /*
152     * although the write manual page doesn't specify this
153     * as a possible errno, it is set when the nfs read
154     * via the mmap'ed file is accessed, so report the
155     * problem as a source access problem, not a target file
156     * problem
157     */
158     if (nbytes < 0) {
159         if (errno == EACCES)
160             perror(srcbuf);
161         else
162             perror(targbuf);
163         (void) close(fi);
164         (void) close(fo);
165         (void) munmap(cp, munmapsize);
166         if (S_ISREG(s2p->st_mode))
167             if (ISREG(*s2p))
168                 (void) unlink(targbuf);
169         if (srcbuf != NULL)
170             free(srcbuf);
171         if (targbuf != NULL)
172             free(targbuf);
173         return (1);
174     }
175     filesize -= nbytes;
176     if (filesize == 0)
177         break;
178     offset += nbytes;
179     if (filesize < mapsize)
180         mapsize = filesize;
181     if (mmap(cp, mapsize, PROT_READ, MAP_SHARED |
182         MAP_FIXED, fi, offset) == MAP_FAILED) {
183         perror(srcbuf);
184         (void) close(fi);
185         (void) close(fo);
186         (void) munmap(cp, munmapsize);
187         if (S_ISREG(s2p->st_mode))
188             if (ISREG(*s2p))
189                 (void) unlink(targbuf);
190         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))
205                 if (ISREG(*s2p))
206                     (void) unlink(targbuf);
207             if (srcbuf != NULL)
208                 free(srcbuf);
209             if (targbuf != NULL)
210                 free(targbuf);
211             return (1);
212         } else if (write(fo, buf, n) != n) {
213             (void) close(fi);
214             (void) close(fo);
215             if (S_ISREG(s2p->st_mode))
216                 if (ISREG(*s2p))
217                     (void) unlink(targbuf);
218             if (srcbuf != NULL)
219                 free(srcbuf);
220             if (targbuf != NULL)
221                 free(targbuf);
222             return (1);
223         }
224     }
225     if (srcbuf != NULL)
226         free(srcbuf);
227     if (targbuf != NULL)
228         free(targbuf);
229     return (0);

```

_____unchanged_portion_omitted_____

```

*****
4951 Sun Mar 1 12:47:16 2015
new/usr/src/lib/libcmdutils/libcmdutils.h
1150 libcmdutils has superfluous #define
*****
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_

```