```
**********************************************************
   50718 Mon Apr 15 19:11:53 2013
new/usr/src/cmd/mv/mv.c
667 cp support for -a flag
**********************************************************
```

```
/*
 * 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 2013 Nexenta Systems, Inc. All rights reserved.
 */

/*
 * Copyright 2009 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.
 */

/*
 * Combined mv/cp/ln command:
 *      mv file1 file2
 *      mv dir1 dir2
 *      mv file1 ... filen dir1
 */
#include <sys/time.h>
#include <signal.h>
#include <locale.h>
#include <stdarg.h>
#include <sys/acl.h>
#include <libcmdutils.h>
#include <aclutils.h>
#include "getresponse.h"

#define FTYPE(A)        (A.st_mode)
#define FMODE(A)        (A.st_mode)
#define UID(A)          (A.st_uid)
```



```
#define GID(A)          (A.st_gid)
#define IDENTICAL(A, B) (A.st_dev == B.st_dev && A.st_ino == B.st_ino)
#define ISDIR(A)        ((A.st_mode & S_IFMT) == S_IFDIR)
#define ISDOOR(A)       ((A.st_mode & S_IFMT) == S_IFDOOR)
#define ISLNK(A)        ((A.st_mode & S_IFMT) == S_IFLNK)
#define ISREG(A)        (((A).st_mode & S_IFMT) == S_IFREG)
#define ISDEV(A)        ((A.st_mode & S_IFMT) == S_IFCHR || \
                        (A.st_mode & S_IFMT) == S_IFBLK || \
                        (A.st_mode & S_IFMT) == S_IFIFO)
#define ISSOCK(A)       ((A.st_mode & S_IFMT) == S_IFSOCK)

#define DELIM   '/'
#define EQ(x, y)        (strcmp(x, y) == 0)
#define FALSE   0
#define MODEBITS (S_ISUID|S_ISGID|S_ISVTX|S_IRWXU|S_IRWXG|S_IRWXO)
#define TRUE 1

static char             *dname(char *);
static int              lnkfil(char *, char *);
static int              cpymve(char *, char *);
static int              chkfiles(char *, char **);
static int              rcopy(char *, char *);
static int              chk_different(char *, char *);
static int              chg_time(char *, struct stat);
static int              chg_mode(char *, uid_t, gid_t, mode_t);
static int              copydir(char *, char *);
static int              copyspecial(char *);
static int              getrealpath(char *, char *);
static void             usage(void);
static void             Perror(char *);
static void             Perror2(char *, char *);
static int              use_stdin(void);
static int              copyattributes(char *, char *);
static int              copy_sysattr(char *, char *);
static tree_node_t      *create_tnode(dev_t, ino_t);

static struct stat      s1, s2, s3, s4;
static int              cpy = FALSE;
static int              mve = FALSE;
static int              lnk = FALSE;
static char             *cmd;
static int              silent = 0;
static int              fflg = 0;
static int              iflg = 0;
static int              pflg = 0;
static int              Rflg = 0;       /* recursive copy */
static int              rflg = 0;       /* recursive copy */
static int              sflg = 0;
static int              Hflg = 0;       /* follow cmd line arg symlink to dir */
static int              Lflg = 0;       /* follow symlinks */
static int              Pflg = 0;       /* do not follow symlinks */
static int              atflg = 0;
static int              attrsilent = 0;
static int              targetexists = 0;
static int              cmdarg;         /* command line argument */
static avl_tree_t       *stree = NULL;  /* source file inode search tree */
static acl_t            *s1acl;
static int              saflg = 0;      /* 'cp' extended system attr. */
static int              srcfd = -1;
static int              targfd = -1;
static int              sourcedirfd = -1;
static int              targetdirfd = -1;
static DIR              *srcdirp = NULL;
static int              srcattrfd = -1;
static int              targattrfd = -1;
static struct stat      attrdir;
```

```
129 /* Extended system attributes support */

131 static int open_source(char  *);
132 static int open_target_srctarg_attrdirs(char  *, char *);
133 static int open_attrdirp(char *);
134 static int traverse_attrfile(struct dirent *, char *, char *, int);
135 static void rewind_attrdir(DIR *);
136 static void close_all();


139 int
140 main(int argc, char *argv[])
141 {
142         int c, i, r, errflg = 0;
143         char target[PATH_MAX];
144         int (*move)(char *, char *);

146         /*
147          * Determine command invoked (mv, cp, or ln)
148          */

150         if (cmd = strrchr(argv[0], '/'))
151                 ++cmd;
152         else
153                 cmd = argv[0];

155         /*
156          * Set flags based on command.
157          */

159         (void) setlocale(LC_ALL, "");
160 #if !defined(TEXT_DOMAIN)       /* Should be defined by cc -D */
161 #define TEXT_DOMAIN "SYS_TEST"  /* Use this only if it weren't */
162 #endif
163         (void) textdomain(TEXT_DOMAIN);
164         if (init_yes() < 0) {
165                 (void) fprintf(stderr, gettext(ERR_MSG_INIT_YES),
166                     strerror(errno));
167                 exit(3);
168         }

170         if (EQ(cmd, "mv"))
171                 mve = TRUE;
172         else if (EQ(cmd, "ln"))
173                 lnk = TRUE;
174         else if (EQ(cmd, "cp"))
175                 cpy = TRUE;
176         else {
177                 (void) fprintf(stderr,
178                     gettext("Invalid command name (%s); expecting "
179                     "mv, cp, or ln.\n"), cmd);
180                 exit(1);
181         }

183         /*
184          * Check for options:
185          *      cp [ -r|-R [-H|-L|-P]] [-afip@/] file1 [file2 ...] target
186          *      cp [-afiprR@/] file1 [file2 ...] target
185          *      cp  -r|-R [-H|-L|-P] [-fip@/] file1 [file2 ...] target
186          *      cp [-fiprR@/] file1 [file2 ...] target
187          *      ln [-f] [-n] [-s] file1 [file2 ...] target
188          *      ln [-f] [-n] [-s] file1 [file2 ...]
189          *      mv [-f|i] file1 [file2 ...] target
190          *      mv [-f|i] dir1 target
191          */
```



```
193         if (cpy) {
194                 while ((c = getopt(argc, argv, "afHiLpPrR@/")) != EOF)
194                 while ((c = getopt(argc, argv, "fHiLpPrR@/")) != EOF)
195                         switch (c) {
196                         case 'f':
197                                 fflg++;
198                                 break;
199                         case 'i':
200                                 iflg++;
201                                 break;
202                         case 'p':
203                                 pflg++;
204 #ifdef XPG4
205                                 attrsilent = 1;
206                                 atflg = 0;
207                                 saflg = 0;
208 #else
209                                 if (atflg == 0)
210                                         attrsilent = 1;
211 #endif
212                                 break;
213                         case 'H':
214                                 /*
215                                  * If more than one of -H, -L, or -P are
216                                  * specified, only the last option specified
217                                  * determines the behavior.
218                                  */
219                                 Lflg = Pflg = 0;
220                                 Hflg++;
221                                 break;
222                         case 'L':
223                                 Hflg = Pflg = 0;
224                                 Lflg++;
225                                 break;
226                         case 'P':
227                                 Lflg = Hflg = 0;
228                                 Pflg++;
229                                 break;
230                         case 'R':
231                                 /*
232                                  * The default behavior of cp -R|-r
233                                  * when specified without -H|-L|-P
234                                  * is -L.
235                                  */
236                                 Rflg++;
237                                 /*FALLTHROUGH*/
238                         case 'r':
239                                 rflg++;
240                                 break;
241                         case 'a':
242                                 Lflg = Hflg = 0;
243                                 pflg++;
244                                 Pflg++;
245                                 Rflg++;
246                                 rflg++;
247                                 break;
248                         case '@':
249                                 atflg++;
250                                 attrsilent = 0;
251 #ifdef XPG4
252                                 pflg = 0;
253 #endif
254                                 break;
255                         case '/':
256                                 saflg++;
```

```
257                                 attrsilent = 0;
258 #ifdef XPG4
259                                 pflg = 0;
260 #endif
261                                 break;
262                         default:
263                                 errflg++;
264                         }
265
266                 /* -R or -r must be specified with -H, -L, or -P */
267                 if ((Hflg || Lflg || Pflg) && !(Rflg || rflg)) {
268                         errflg++;
269                 }
270
271         } else if (mve) {
272                 while ((c = getopt(argc, argv, "fis")) != EOF)
273                         switch (c) {
274                         case 'f':
275                                 silent++;
276 #ifdef XPG4
277                                 iflg = 0;
278 #endif
279                                 break;
280                         case 'i':
281                                 iflg++;
282 #ifdef XPG4
283                                 silent = 0;
284 #endif
285                                 break;
286                         default:
287                                 errflg++;
288                         }
289         } else { /* ln */
290                 while ((c = getopt(argc, argv, "fns")) != EOF)
291                         switch (c) {
292                         case 'f':
293                                 silent++;
294                                 break;
295                         case 'n':
296                                 /* silently ignored; this is the default */
297                                 break;
298                         case 's':
299                                 sflg++;
300                                 break;
301                         default:
302                                 errflg++;
303                         }
304         }
305
306         /*
307          * For BSD compatibility allow - to delimit the end of
308          * options for mv.
309          */
310         if (mve && optind < argc && (strcmp(argv[optind], "-") == 0))
311                 optind++;
312
313         /*
314          * Check for sufficient arguments
315          * or a usage error.
316          */
317
318         argc -= optind;
319         argv  = &argv[optind];
320
321         if ((argc < 2 && lnk != TRUE) || (argc < 1 && lnk == TRUE)) {
322                 (void) fprintf(stderr,
```



```
323                     gettext("%s: Insufficient arguments (%d)\n"),
324                     cmd, argc);
325                 usage();
326         }
327
328         if (errflg != 0)
329                 usage();
330
331         /*
332          * If there is more than a source and target,
333          * the last argument (the target) must be a directory
334          * which really exists.
335          */
336
337         if (argc > 2) {
338                 if (stat(argv[argc-1], &s2) < 0) {
339                         (void) fprintf(stderr,
340                             gettext("%s: %s not found\n"),
341                             cmd, argv[argc-1]);
342                         exit(2);
343                 }
344
345                 if (!ISDIR(s2)) {
346                         (void) fprintf(stderr,
347                             gettext("%s: Target %s must be a directory\n"),
348                             cmd, argv[argc-1]);
349                         usage();
350                 }
351         }
352
353         if (strlen(argv[argc-1]) >= PATH_MAX) {
354                 (void) fprintf(stderr,
355                     gettext("%s: Target %s file name length exceeds PATH_MAX"
356                     " %d\n"), cmd, argv[argc-1], PATH_MAX);
357                 exit(78);
358         }
359
360         if (argc == 1) {
361                 if (!lnk)
362                         usage();
363                 (void) strcpy(target, ".");
364         } else {
365                 (void) strcpy(target, argv[--argc]);
366         }
367
368         /*
369          * Perform a multiple argument mv|cp|ln by
370          * multiple invocations of cpymve() or lnkfil().
371          */
372         if (lnk)
373                 move = lnkfil;
374         else
375                 move = cpymve;
376
377         r = 0;
378         for (i = 0; i < argc; i++) {
379                 stree = NULL;
380                 cmdarg = 1;
381                 r += move(argv[i], target);
382         }
383
384         /*
385          * Show errors by nonzero exit code.
386          */
387
388         return (r?2:0);
```

```
 389 }
______unchanged_portion_omitted_

1307 static void
1308 usage(void)
1309 {
1310         /*
1311          * Display usage message.
1312          */

1314         if (mve) {
1315                 (void) fprintf(stderr, gettext(
1316                     "Usage: mv [-f] [-i] f1 f2\n"
1317                     "       mv [-f] [-i] f1 ... fn d1\n"
1318                     "       mv [-f] [-i] d1 d2\n"));
1319         } else if (lnk) {
1320 #ifdef XPG4
1321                 (void) fprintf(stderr, gettext(
1322                     "Usage: ln [-f] [-s] f1 [f2]\n"
1323                     "       ln [-f] [-s] f1 ... fn d1\n"
1324                     "       ln [-f] -s d1 d2\n"));
1325 #else
1326                 (void) fprintf(stderr, gettext(
1327                     "Usage: ln [-f] [-n] [-s] f1 [f2]\n"
1328                     "       ln [-f] [-n] [-s] f1 ... fn d1\n"
1329                     "       ln [-f] [-n] -s d1 d2\n"));
1330 #endif
1331         } else if (cpy) {
1332                 (void) fprintf(stderr, gettext(
1333                     "Usage: cp [-a] [-f] [-i] [-p] [-@] [-/] f1 f2\n"
1334                     "       cp [-a] [-f] [-i] [-p] [-@] [-/] f1 ... fn d1\n"
1335                     "       cp [-r|-R [-H|-L|-P]] [-a] [-f] [-i] [-p] [-@] "
1336                     "[-/] d1 ... dn-1 dn\n"));
1326                     "Usage: cp [-f] [-i] [-p] [-@] [-/] f1 f2\n"
1327                     "       cp [-f] [-i] [-p] [-@] [-/] f1 ... fn d1\n"
1328                     "       cp -r|-R [-H|-L|-P] [-f] [-i] [-p] [-@] [-/] "
1329                     "d1 ... dn-1 dn\n"));
1337         }
1338         exit(2);
1339 }
______unchanged_portion_omitted_
```

```
**********************************************************
   14126 Mon Apr 15 19:11:53 2013
new/usr/src/man/man1/cp.1
667 cp support for -a flag
**********************************************************
'\" te
.\" Copyright 2013 Nexenta Systems, Inc. All rights reserved.
.\" Copyright (c) 1992, X/Open Company Limited All Rights Reserved
.\" Copyright 1989 AT&T
.\" Portions Copyright (c) 2007, Sun Microsystems, Inc. All Rights Reserved
.\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
.\" http://www.opengroup.org/bookstore/.
.\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
.\"  This notice shall appear on any product containing this material.
.\" The contents of this file are subject to the terms of the Common Development
.\" You can obtain a copy of the license at usr/src/OPENSOLARIS.LICENSE or http:
.\" When distributing Covered Code, include this CDDL HEADER in each file and in
.TH CP 1 "Apr 15, 2013"
.TH CP 1 "Oct 30, 2007"
.SH NAME
cp \- copy files
.SH SYNOPSIS
.LP
.nf
\fB/usr/bin/cp\fR [\fB-afip@/\fR] \fIsource_file\fR \fItarget_file\fR
\fB/usr/bin/cp\fR [\fB-fip@/\fR] \fIsource_file\fR \fItarget_file\fR
.fi

.LP
.nf
\fB/usr/bin/cp\fR [\fB-afip@/\fR] \fIsource_file\fR... \fItarget\fR
\fB/usr/bin/cp\fR [\fB-fip@/\fR] \fIsource_file\fR... \fItarget\fR
.fi

.LP
.nf
\fB/usr/bin/cp\fR [\fB-r\fR | \fB-R\fR [\fB-H\fR | \fB-L\fR | \fB-P\fR]] [\fB-af
\fB/usr/bin/cp\fR \fB-r\fR | \fB-R\fR [\fB-H\fR | \fB-L\fR | \fB-P\fR] [\fB-fip@
.fi

.LP
.nf
\fB/usr/bin/cp\fR [\fB-R\fR | \fB-R\fR [\fB-H\fR | \fB-L\fR | \fB-P\fR]] [\fB-af
\fB/usr/bin/cp\fR \fB-R\fR | \fB-R\fR [\fB-H\fR | \fB-L\fR | \fB-P\fR] [\fB-fip@
.fi

.LP
.nf
\fB/usr/xpg4/bin/cp\fR [\fB-afip@/\fR] \fIsource_file\fR \fItarget_file\fR
\fB/usr/xpg4/bin/cp\fR [\fB-fip@/\fR] \fIsource_file\fR \fItarget_file\fR
.fi

.LP
.nf
\fB/usr/xpg4/bin/cp\fR [\fB-afip@/\fR] \fIsource_file\fR... \fItarget\fR
\fB/usr/xpg4/bin/cp\fR [\fB-fip@/\fR] \fIsource_file\fR... \fItarget\fR
.fi

.LP
.nf
\fB/usr/xpg4/bin/cp\fR [\fB-r\fR | \fB-R\fR [\fB-H\fR | \fB-L\fR | \fB-P\fR]] [\
\fB/usr/xpg4/bin/cp\fR \fB-r\fR | \fB-R\fR [\fB-H\fR | \fB-L\fR | \fB-P\fR] [\fB
.fi

.LP
.nf
```



```
\fB/usr/xpg4/bin/cp\fR [\fB-R\fR | \fB-R\fR [\fB-H\fR | \fB-L\fR | \fB-P\fR]] [\
\fB/usr/xpg4/bin/cp\fR \fB-R\fR | \fB-R\fR [\fB-H\fR | \fB-L\fR | \fB-P\fR] [\fB
.fi

.SH DESCRIPTION
.sp
.LP
In the first synopsis form, neither \fIsource_file\fR nor \fItarget_file\fR are
directory files, nor can they have the same name. The \fBcp\fR utility copies
the contents of \fIsource_file\fR to the destination path named by
\fItarget_file\fR. If \fItarget_file\fR exists, \fBcp\fR overwrites its
contents, but the mode (and \fBACL\fR if applicable), owner, and group
associated with it are not changed. The last modification time of
\fItarget_file\fR and the last access time of \fIsource_file\fR are set to the
time the copy was made. If \fItarget_file\fR does not exist, \fBcp\fR creates a
new file named \fItarget_file\fR that has the same mode as \fIsource_file\fR
except that the sticky bit is not set unless the user is super-user. In this
case, the owner and group of \fItarget_file\fR are those of the user, unless
the setgid bit is set on the directory containing the newly created file. If
the directory's setgid bit is set, the newly created file has the group of the
containing directory rather than of the creating user. If \fItarget_file\fR is
a link to another file, \fBcp\fR overwrites the link destination with the
contents of \fIsource_file\fR; the link(s) from \fItarget_file\fR remains.
.sp
.LP
In the second synopsis form, one or more \fIsource_file\fRs are copied to the
directory specified by \fItarget\fR. It is an error if any \fIsource_file\fR is
a file of type directory, if \fItarget\fR either does not exist or is not a
directory.
.sp
.LP
In the third or fourth synopsis forms, one or more directories specified by
\fIsource_dir\fR are copied to the directory specified by \fItarget\fR. Either
the \fB-r\fR or \fB-R\fR must be specified. For each \fIsource_dir\fR, \fBcp\fR
copies all files and subdirectories.
.SH OPTIONS
.sp
.LP
The following options are supported for both \fB/usr/bin/cp\fR and
\fB/usr/xpg4/bin/cp\fR:
.sp
.ne 2
.na
\fB\fB-a\fR\fR
.ad
.RS 6n
Archive mode. Same as -RpP.
.RE

.sp
.ne 2
.na
\fB\fB-f\fR\fR
.ad
.RS 6n
Unlink. If a file descriptor for a destination file cannot be obtained, this
option attempts to unlink the destination file and proceed.
.RE

.sp
.ne 2
.na
\fB\fB-H\fR\fR
.ad
.RS 6n
Takes actions based on the type and contents of the file referenced by any
```

```
119 symbolic link specified as a \fIsource_file\fR operand.
120 .sp
121 If the \fIsource_file\fR operand is a symbolic link, then \fBcp\fR copies the
122 file referenced by the symbolic link for the \fIsource_file\fR operand. All
123 other symbolic links encountered during traversal of a file hierarchy are
124 preserved.
125 .RE

127 .sp
128 .ne 2
129 .na
130 \fB\fB-i\fR\fR
131 .ad
132 .RS 6n
133 Interactive. \fBcp\fR prompts for confirmation whenever the copy would
134 overwrite an existing \fItarget\fR. An affirmative response means that the copy
135 should proceed. Any other answer prevents \fBcp\fR from overwriting
136 \fItarget\fR.
137 .RE

139 .sp
140 .ne 2
141 .na
142 \fB\fB-L\fR\fR
143 .ad
144 .RS 6n
145 Takes actions based on the type and contents of the file referenced by any
146 symbolic link specified as a \fIsource_file\fR operand or any symbolic links
147 encountered during traversal of a file hierarchy.
148 .sp
149 Copies files referenced by symbolic links. Symbolic links encountered during
150 traversal of a file hierarchy are not preserved.
151 .RE

153 .sp
154 .ne 2
155 .na
156 \fB\fB-p\fR\fR
157 .ad
158 .RS 6n
159 Preserve. The \fBcp\fR utility duplicates not only the contents of
160 \fIsource_file\fR, but also attempts to preserve its ACL, access and
161 modification times, extended attributes, extended system attributes, file mode,
162 and owner and group ids.
163 .sp
164 If \fBcp\fR is unable to preserve the access and modification times, extended
165 attributes, or the file mode, \fBcp\fR does not consider it a failure. If
166 \fBcp\fR is unable to preserve the owner and group id, the copy does not fail,
167 but \fBcp\fR silently clears the \fBS_ISUID\fR and \fBS_ISGID\fR bits from the
168 file mode of the target. The copy fails if \fBcp\fR is unable to clear these
169 bits. If \fBcp\fR is unable to preserve the ACL or extended system attributes,
170 the copy fails. If the copy fails, then a diagnostic message is written to
171 \fBstderr\fR and (after processing any remaining operands) \fBcp\fR exits with
172 a \fBnon-zero\fR exit status.
173 .RE

175 .sp
176 .ne 2
177 .na
178 \fB\fB-P\fR\fR
179 .ad
180 .RS 6n
181 Takes actions on any symbolic link specified as a \fIsource_file\fR operand or
182 any symbolic link encountered during traversal of a file hierarchy.
183 .sp
184 Copies symbolic links. Symbolic links encountered during traversal of a file
```



```
185 hierarchy are preserved.
186 .RE

188 .sp
189 .ne 2
190 .na
191 \fB\fB-r\fR\fR
192 .ad
193 .RS 6n
194 Recursive. \fBcp\fR copies the directory and all its files, including any
195 subdirectories and their files to \fItarget\fR. Unless the \fB-H\fR, \fB-L\fR,
196 or \fB-P\fR option is specified, the \fB-L\fR option is used as the default
197 mode.
198 .RE

200 .sp
201 .ne 2
202 .na
203 \fB\fB-R\fR\fR
204 .ad
205 .RS 6n
206 Same as \fB-r\fR, except pipes are replicated, not read from.
207 .RE

209 .sp
210 .ne 2
211 .na
212 \fB\fB-@\fR\fR
213 .ad
214 .RS 6n
215 Preserves extended attributes. \fBcp\fR attempts to copy all of the source
216 file's extended attributes along with the file data to the destination file.
217 .RE

219 .sp
220 .ne 2
221 .na
222 \fB\fB-/\fR\fR
223 .ad
224 .RS 6n
225 Preserves extended attributes and extended system attributes. Along with the
226 file's data, the \fBcp\fR utility attempts to copy extended attributes and
227 extended system attributes from each source file, and extended system
228 attributes associated with extended attributes to the destination file. If
229 \fBcp\fR is unable to copy extended attributes or extended system attributes,
230 then a diagnostic message is written to \fBstderr\fR and (after processing any
231 remaining operands) exits with a \fBnon-zero\fR exit status.
232 .RE

234 .sp
235 .LP
236 Specifying more than one of the mutually-exclusive options \fB-H\fR, \fB-L\fR,
237 and \fB-P\fR is not considered an error. The last option specified determines
238 the behavior of the utility.
239 .SS "/usr/bin/cp"
240 .sp
241 .LP
242 If the \fB-p\fR option is specified with either the \fB-@\fR option or the
243 \fB-/\fR option, \fB/usr/bin/cp\fR behaves as follows
244 .RS +4
245 .TP
246 .ie t \(bu
247 .el o
248 When both \fB-p\fR and \fB-@\fR are specified in any order, the copy fails if
249 extended attributes cannot be copied.
250 .RE
```

```
.RS +4
.TP
.ie t \(bu
.el o
When both \fB-p\fR and \fB-/\fR are specified in any order, the copy fails if
extended system attributes cannot be copied.
.RE
.SS "/usr/xpg4/bin/cp"
.sp
.LP
If the \fB-p\fR option is specified with either the \fB-@\fR option or the
\fB-/\fR option, /\fBusr/xpg4/bin/cp\fR behaves as follows:
.RS +4
.TP
.ie t \(bu
.el o
When both \fB-p\fR and \fB-@\fR are specified, the last option specified
determines whether the copy fails if extended attributes cannot be preserved.
.RE
.RS +4
.TP
.ie t \(bu
.el o
When both \fB-p\fR and \fB-/\fR are specified, the last option specified
determines whether the copy fails if extended system attributes cannot be
preserved.
.RE
.SH OPERANDS
.sp
.LP
The following operands are supported:
.sp
.ne 2
.na
\fB\fIsource_file\fR\fR
.ad
.RS 15n
A pathname of a regular file to be copied.
.RE

.sp
.ne 2
.na
\fB\fIsource_dir\fR\fR
.ad
.RS 15n
A pathname of a directory to be copied.
.RE

.sp
.ne 2
.na
\fB\fItarget_file\fR\fR
.ad
.RS 15n
A pathname of an existing or non-existing file, used for the output when a
single file is copied.
.RE

.sp
.ne 2
.na
\fB\fItarget\fR\fR
.ad
.RS 15n
A pathname of a directory to contain the copied files.
```



```
.RE

.SH USAGE
.sp
.LP
See \fBlargefile\fR(5) for the description of the behavior of \fBcp\fR when
encountering files greater than or equal to 2 Gbyte ( 2^31 bytes).
.SH EXAMPLES
.LP
\fBExample 1 \fRCopying a File
.sp
.LP
The following example copies a file:

.sp
.in +2
.nf
example% cp goodies goodies.old

example% ls goodies*
goodies goodies.old
.fi
.in -2
.sp

.LP
\fBExample 2 \fRCopying a List of Files
.sp
.LP
The following example copies a list of files to a destination directory:

.sp
.in +2
.nf
example% cp ~/src/* /tmp
.fi
.in -2
.sp

.LP
\fBExample 3 \fRCopying a Directory
.sp
.LP
The following example copies a directory, first to a new, and then to an
existing destination directory

.sp
.in +2
.nf
example% ls ~/bkup
/usr/example/fred/bkup not found

example% cp \fB-r\fR ~/src ~/bkup

example% ls \fB-R\fR ~/bkup
x.c y.c z.sh

example% cp \fB-r\fR ~/src ~/bkup

example% ls \fB-R\fR ~/bkup
src x.c y.c z.sh
src:
x.c y.c z.s
.fi
.in -2
.sp
```

```
.LP
\fBExample 4 \fRCopying Extended File System Attributes
.sp
.LP
The following example copies extended file system attributes:

.sp
.in +2
.nf
$ ls -/ c file1
-rw-r--r--   1 foo   staff          0 Oct 29 20:04 file1
                {AH-----m--}

$ cp -/ file1 file2
$ ls -/c file2
-rw-r--r--   1 foo  staff          0 Oct 29 20:17 file2
                {AH-----m--}
.fi
.in -2
.sp

.LP
\fBExample 5 \fRFailing to Copy Extended System Attributes
.sp
.LP
The following example fails to copy extended system attributes:

.sp
.in +2
.nf
$ ls -/c file1
-rw-r--r--   1 foo    staff          0 Oct 29 20:04 file1
                {AH-----m--}

$ cp -/ file1 /tmp
cp: Failed to copy extended system attributes from file1 to /tmp/file1


$ ls -/c /tmp/file1
-rw-r--r--   1 foo    staff          0 Oct 29 20:09 /tmp/file1
                {}
.fi
.in -2
.sp

.SH ENVIRONMENT VARIABLES
.sp
.LP
See \fBenviron\fR(5) for descriptions of the following environment variables
that affect the execution of \fBcp\fR: \fBLANG\fR, \fBLC_ALL\fR,
\fBLC_COLLATE\fR, \fBLC_CTYPE\fR, \fBLC_MESSAGES\fR, and \fBNLSPATH\fR.
.sp
.LP
Affirmative responses are processed using the extended regular expression
defined for the \fByesexpr\fR keyword in the \fBLC_MESSAGES\fR category of the
user's locale. The locale specified in the \fBLC_COLLATE\fR category defines
the behavior of ranges, equivalence classes, and multi-character collating
elements used in the expression defined for \fByesexpr\fR. The locale specified
in \fBLC_CTYPE\fR determines the locale for interpretation of sequences of
bytes of text data a characters, the behavior of character classes used in the
expression defined for the \fByesexpr\fR. See \fBlocale\fR(5).
.SH EXIT STATUS
.sp
.LP
The following exit values are returned:
```



```
.sp
.ne 2
.na
\fB\fB0\fR\fR
.ad
.RS 6n
All files were copied successfully.
.RE

.sp
.ne 2
.na
\fB\fB>0\fR\fR
.ad
.RS 6n
An error occurred.
.RE

.SH ATTRIBUTES
.sp
.LP
See \fBattributes\fR(5) for descriptions of the following attributes:
.SS "/usr/bin/cp"
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
CSI     Enabled
_
Interface Stability     Committed
.TE

.SS "/usr/xpg4/bin/cp"
.sp

.sp
.TS
box;
c | c
l | l .
ATTRIBUTE TYPE  ATTRIBUTE VALUE
_
CSI     Enabled
_
Interface Stability     Committed
.TE

.SH SEE ALSO
.sp
.LP
\fBchmod\fR(1), \fBchown\fR(1), \fBsetfacl\fR(1), \fButime\fR(2),
\fBfgetattr\fR(3C), \fBattributes\fR(5), \fBenviron\fR(5), \fBfsattr\fR(5),
\fBlargefile\fR(5), \fBlocale\fR(5), \fBstandards\fR(5)
.SH NOTES
.sp
.LP
The permission modes of the source file are preserved in the copy.
.sp
.LP
A \fB--\fR permits the user to mark the end of any command line options
explicitly, thus allowing \fBcp\fR to recognize filename arguments that begin
```

```
 515 with a \fB-\fR.
```