```
**********************************************************
  241505 Sun Aug  5 16:35:34 2012
new/usr/src/cmd/cpio/cpio.c
1154 cpio needs a quiet option
**********************************************************
```

```
   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 (c) 1988, 2010, Oracle and/or its affiliates. All rights reserved.
  23  * Copyright 2012 Milan Jurik. All rights reserved.
  24  * Copyright (c) 2012 Gary Mills
  25  */

  27 /*      Copyright (c) 1983, 1984, 1985, 1986, 1987, 1988, 1989 AT&T     */
  28 /*      All Rights Reserved                                             */

  30 /*
  31  * Portions of this source code were derived from Berkeley 4.3 BSD
  32  * under license from the Regents of the University of California.
  33  */

  35 #include <stdio.h>
  36 #include <sys/types.h>
  37 #include <errno.h>
  38 #include <unistd.h>
  39 #include <stdlib.h>
  40 #include <fcntl.h>
  41 #include <memory.h>
  42 #include <string.h>
  43 #include <stdarg.h>
  44 #include <sys/stat.h>
  45 #include <sys/statvfs.h>
  46 #include <sys/mkdev.h>
  47 #include <sys/param.h>
  48 #include <utime.h>
  49 #include <pwd.h>
  50 #include <grp.h>
  51 #include <signal.h>
  52 #include <ctype.h>
  53 #include <locale.h>
  54 #include <sys/ioctl.h>
  55 #include <sys/mtio.h>
  56 #include <sys/fdio.h>
  57 #include "cpio.h"
  58 #include <sys/acl.h>
  59 #include <sys/time.h>
  60 #include <sys/resource.h>
  61 #include <fnmatch.h>
```



```
  62 #include <libgen.h>
  63 #include <libintl.h>
  64 #include <dirent.h>
  65 #include <limits.h>
  66 #include <aclutils.h>
  67 #if defined(_PC_SATTR_ENABLED)
  68 #include <libnvpair.h>
  69 #include <attr.h>
  70 #include <libcmdutils.h>
  71 #endif  /* _PC_SATTR_ENABLED */
  72 #ifdef SOLARIS_PRIVS
  73 #include <priv.h>
  74 #endif  /* SOLARIS_PRIVS */

  76 /*
  77  * Special kludge for off_t being a signed quantity.
  78  */
  79 #if _FILE_OFFSET_BITS == 64
  80 typedef u_longlong_t    u_off_t;
  81 #else
  82 typedef ulong_t         u_off_t;
  83 #endif

  85 #define SECMODE 0xe080

  87 #define DEVNULL         "/dev/null"
  88 #define XATTRHDR        ".hdr"

  90 #define NAMELEN         32
  91 #define TYPELEN         16
  92 #define PERMLEN         4

  94 #define FILE_COPIED     1
  95 #define FILE_LINKED     2
  96 #define FILE_PASS_ERR   -1

  98 #define ARCHIVE_NORMAL  0
  99 #define ARCHIVE_ACL     1
 100 #define ARCHIVE_XATTR   2
 101 #define ARCHIVE_SPARSE  3

 103 #ifndef VIEW_READONLY
 104 #define VIEW_READONLY   "SUNWattr_ro"
 105 #endif

 107 #ifndef VIEW_READWRITE
 108 #define VIEW_READWRITE  "SUNWattr_rw"
 109 #endif


 112 #define LSTAT(dir, path, statbuf) fstatat(dir, \
 113     get_component((Gen.g_attrnam_p == NULL) ? \
 114     path : Gen.g_attrnam_p), statbuf, AT_SYMLINK_NOFOLLOW)
 115 #define STAT(dir, path, statbuf) fstatat(dir, \
 116     get_component((Gen.g_attrnam_p == NULL) ? \
 117     path : Gen.g_attrnam_p), statbuf, 0)

 119 /*
 120  *      These limits reflect the maximum size regular file that
 121  *      can be archived, depending on the archive type. For archives
 122  *      with character-format headers (odc, tar, ustar) we use
 123  *      CHAR_OFFSET_MAX.  For archives with SVR4 ASCII headers (-c, -H crc)
 124  *      we store filesize in an 8-char hexadecimal string and use
 125  *      ASC_OFFSET_MAX.  Otherwise, we are limited to the size that will
 126  *      fit in a signed long value.
 127  */
```

```
128 #define CHAR_OFFSET_MAX 077777777777ULL /* 11 octal digits */
129 #define ASC_OFFSET_MAX  0XFFFFFFFF      /* 8 hexadecimal digits */
130 #define BIN_OFFSET_MAX  LONG_MAX        /* signed long max value */

132 #define POSIXMODES      07777

134 static char     aclchar = ' ';

136 static struct Lnk *add_lnk(struct Lnk **);
137 static int bfill(void);
138 static void bflush(void);
139 static int chgreel(int dir);
140 static int ckname(int);
141 static void ckopts(long mask);
142 static long cksum(char hdr, int byt_cnt, int *err);
143 static int creat_hdr(void);
144 static int creat_lnk(int dirfd, char *name1_p, char *name2_p);
145 static int creat_spec(int dirfd);
146 static int creat_tmp(char *nam_p);
147 static void data_in(int proc_mode);
148 static void data_out(void);
149 static void data_pass(void);
150 static void file_in(void);
151 static int file_out(void);
152 static int file_pass(void);
153 static void flush_lnks(void);
154 static int gethdr(void);
155 static int getname(void);
156 static void getpats(int largc, char **largv);
157 static void ioerror(int dir);
158 static int matched(void);
159 static int missdir(char *nam_p);
160 static long mklong(short v[]);
161 static void mkshort(short sval[], long v);
162 static int openout(int dirfd);
163 static int read_hdr(int hdr);
164 static void reclaim(struct Lnk *l_p);
165 static void rstbuf(void);
166 static void setpasswd(char *nam);
167 static void rstfiles(int over, int dirfd);
168 static void scan4trail(void);
169 static void setup(int largc, char **largv);
170 static void set_tym(int dirfd, char *nam_p, time_t atime, time_t mtime);
171 static void sigint(int sig);
172 static void swap(char *buf_p, int cnt);
173 static void usage(void);
174 static void verbose(char *nam_p);
175 static void write_hdr(int arcflag, off_t len);
176 static void write_trail(void);
177 static int ustar_dir(void);
178 static int ustar_spec(void);
179 static struct stat *convert_to_old_stat(struct stat *, char *, char *);
180 static void read_bar_vol_hdr(void);
181 static void read_bar_file_hdr(void);
182 static void setup_uncompress(FILE **);
183 static void skip_bar_volhdr(void);
184 static void bar_file_in(void);
185 static int g_init(int *devtype, int *fdes);
186 static int g_read(int, int, char *, unsigned);
187 static int g_write(int, int, char *, unsigned);
188 static int is_floppy(int);
189 static int is_tape(int);
190 static void write_ancillary(char *buf, size_t len, boolean_t padding);
191 static int remove_dir(char *);
192 static int save_cwd(void);
193 static void rest_cwd(int cwd);
```



```
195 static void xattrs_out(int (*func)());
196 static void get_parent(char *path, char *dir);
197 static void prepare_xattr_hdr(char **attrbuf, char *filename,
198     char *attrname, char typeflag, struct Lnk *linkinfo, int *rlen);
199 static char tartype(int type);
200 static int openfile(int omode);
201 static mode_t attrmode(char type);
202 static char *get_component(char *path);
203 static int open_dir(char *name);
204 static int open_dirfd();
205 static void close_dirfd();
206 static void write_xattr_hdr();
207 static char *skipslashes(char *string, char *start);
208 static int read_xattr_hdr();
209 static void chop_endslashes(char *path);


212 /* helpful types */

214 static
215 struct passwd   *Curpw_p,       /* Current password entry for -t option */
216                 *Rpw_p,         /* Password entry for -R option */
217                 *dpasswd;

219 static
220 struct group    *Curgr_p,       /* Current group entry for -t option */
221                 *dgroup;

223 /* Data structure for buffered I/O. */

225 static
226 struct buf_info {
227         char    *b_base_p,      /* Pointer to base of buffer */
228                 *b_out_p,       /* Position to take bytes from buffer at */
229                 *b_in_p,        /* Position to put bytes into buffer at */
230                 *b_end_p;       /* Pointer to end of buffer */
231         long    b_cnt,          /* Count of unprocessed bytes */
232                 b_size;         /* Size of buffer in bytes */
233 } Buffr;
_____unchanged_portion_omitted_
540 #endif


543 /*
544  *
545  * cpio has been changed to support extended attributes.
546  *
547  * As part of this change cpio has been changed to use the new *at() syscalls
548  * such as openat, fchownat(), unlinkat()...
549  *
550  * This was done so that attributes can be handled with as few code changes
551  * as possible.
552  *
553  * What this means is that cpio now opens the directory that a file or directory
554  * resides in and then performs *at() functions to manipulate the entry.
555  *
556  * For example a new file is now created like this:
557  *
558  * dfd = open(<some dir path>)
559  * fd = openat(dfd, <name>,....);
560  *
561  * or in the case of an extended attribute
562  *
563  * dfd = attropen(<pathname>, ".", ....)
564  *
```

```
 565  * Once we have a directory file descriptor all of the *at() functions can
 566  * be applied to it.
 567  *
 568  * unlinkat(dfd, <component name>,...)
 569  * fchownat(dfd, <component name>,..)
 570  *
 571  * This works for both normal namespace files and extended attribute file
 572  *
 573  */

 575 /*
 576  * Extended attribute layout
 577  *
 578  * Extended attributes are stored in two pieces.
 579  * 1. An attribute header which has information about
 580  *    what file the attribute is for and what the attribute
 581  *    is named.
 582  * 2. The attribute record itself.  Stored as a normal file type
 583  *    of entry.
 584  * Both the header and attribute record have special modes/typeflags
 585  * associated with them.
 586  *
 587  * The names of the header in the archive look like:
 588  * /dev/null/attr.hdr
 589  *
 590  * The name of the attribute looks like:
 591  * /dev/null/attr.
 592  *
 593  * This is done so that an archiver that doesn't understand these formats
 594  * can just dispose of the attribute records unless the user chooses to
 595  * rename them via cpio -r or pax -i
 596  *
 597  * The format is composed of a fixed size header followed
 598  * by a variable sized xattr_buf. If the attribute is a hard link
 599  * to another attribute, then another xattr_buf section is included
 600  * for the link.
 601  *
 602  * The xattr_buf is used to define the necessary "pathing" steps
 603  * to get to the extended attribute.  This is necessary to support
 604  * a fully recursive attribute model where an attribute may itself
 605  * have an attribute.
 606  *
 607  * The basic layout looks like this.
 608  *
 609  *     --------------------------------
 610  *     |                              |
 611  *     |         xattr_hdr            |
 612  *     |                              |
 613  *     --------------------------------
 614  *     --------------------------------
 615  *     |                              |
 616  *     |        xattr_buf             |
 617  *     |                              |
 618  *     --------------------------------
 619  *     --------------------------------
 620  *     |                              |
 621  *     |      (optional link info)    |
 622  *     |                              |
 623  *     --------------------------------
 624  *     --------------------------------
 625  *     |                              |
 626  *     |      attribute itself        |
 627  *     |      stored as normal tar    |
 628  *     |      or cpio data with       |
 629  *     |      special mode or         |
 630  *     |      typeflag                |
```



```
 631  *     |                              |
 632  *     --------------------------------
 633  *
 634  */

 636 /*
 637  * Extended attributes structures
 638  *
 639  * xattrhead is the complete extended attribute header, as read of off
 640  * disk/tape. It includes the variable xattr_buf portion.
 641  *
 642  * xattrp is basically an offset into xattrhead that points to the
 643  * "pathing" section which defines how to get to the attribute.
 644  *
 645  * xattr_linkp is identical to xattrp except that it is used for linked
 646  * attributes.  It provides the pathing steps to get to the linked
 647  * attribute.
 648  *
 649  * These structures are updated when an extended attribute header is read off
 650  * of disk/tape.
 651  */
 652 static struct xattr_hdr *xattrhead;
 653 static struct xattr_buf *xattrp;
 654 static struct xattr_buf *xattr_linkp;
 655 static int              xattrbadhead;   /* is extended attribute header bad? */

 657 static int      append_secattr(char **, int *, acl_t *);

 659 /*
 660  * Note regarding cpio and changes to ensure cpio doesn't try to second
 661  * guess whether it runs with sufficient privileges or not:
 662  *
 663  * cpio has been changed so that it doesn't carry a second implementation of
 664  * the kernel's policy with respect to privileges.  Instead of attempting
 665  * to restore uid and gid from an archive only if cpio is run as uid 0,
 666  * cpio now *always* tries to restore the uid and gid from the archive
 667  * except when the -R option is specified.  When the -R is specified,
 668  * the uid and gid of the restored file will be changed to those of the
 669  * login id specified.  In addition, chown(), set_tym(), and chmod() should
 670  * only be executed once during archive extraction, and to ensure
 671  * setuid/setgid bits are restored properly, chown() should always be
 672  * executed before chmod().
 673  *
 674  * Note regarding debugging mechanism for cpio:
 675  *
 676  * The following mechanism is provided to allow us to debug cpio in complicated
 677  * situations, like when it is part of a pipe.  The idea is that you compile
 678  * with -DWAITAROUND defined, and then add the "-z" command line option to the
 679  * target cpio invocation.  If stderr is available, it will tell you to which
 680  * pid to attach the debugger; otherwise, use ps to find it.  Attach to the
 681  * process from the debugger, and, *PRESTO*, you are there!
 682  *
 683  * Simply assign "waitaround = 0" once you attach to the process, and then
 684  * proceed from there as usual.
 685  */

 687 #ifdef WAITAROUND
 688 int waitaround = 0;             /* wait for rendezvous with the debugger */
 689 #endif

 691 #define EXIT_CODE       (Error_cnt > 255 ? 255 : Error_cnt)

 693 /*
 694  * main: Call setup() to process options and perform initializations,
 695  * and then select either copy in (-i), copy out (-o), or pass (-p) action.
 696  */
```

```
698 int
699 main(int argc, char **argv)
700 {
701         int i;
702         int passret;

704         (void) setlocale(LC_ALL, "");
705 #if !defined(TEXT_DOMAIN)       /* Should be defined by cc -D */
706 #define TEXT_DOMAIN "SYS_TEST"  /* Use this only if it weren't */
707 #endif
708         (void) textdomain(TEXT_DOMAIN);

710         (void) memset(&Gen, 0, sizeof (Gen));
711         myname = e_strdup(E_EXIT, basename(argv[0]));
712         setup(argc, argv);

714         if (signal(SIGINT, sigint) == SIG_IGN)
715                 (void) signal(SIGINT, SIG_IGN);
716         switch (Args & (OCi | OCo | OCp)) {
717         case OCi: /* COPY IN */
718                 Hdr_type = NONE;
719                 if (Atflag || SysAtflag) {
720                         /*
721                          * Save the current working directory, so
722                          * we can change back here after cd'ing into
723                          * the attribute directory when processing
724                          * attributes.
725                          */
726                         if ((attr_baseparent_fd = save_cwd()) < 0) {
727                                 msg(EXT, "Unable to open current directory.");
728                         }
729                 }
730                 while ((i = gethdr()) != 0) {
731                         Gen.g_dirfd = -1;
732                         if (i == 1) {
733                                 file_in();
734                                 /*
735                                  * Any ACL info for this file would or should
736                                  * have been used after file_in(); clear out
737                                  * aclp so it is is not erroneously used on
738                                  * the next file.
739                                  */
740                                 if (aclp != NULL) {
741                                         acl_free(aclp);
742                                         aclp = NULL;
743                                 }
744                                 acl_is_set = 0;
745                         }
746                         (void) memset(&Gen, 0, sizeof (Gen));
747                 }
748                 /* Do not count "extra" "read-ahead" buffered data */
749                 if (Buffr.b_cnt > Bufsize)
750                         Blocks -=  (u_longlong_t)(Buffr.b_cnt / Bufsize);
751                 break;
752         case OCo: /* COPY OUT */
753                 if (Args & OCA) {
754                         scan4trail();
755                 }

757                 Gen.g_dirfd = -1;
758                 Gen.g_dirpath = NULL;
759                 sl_preview_synonyms();

761                 while ((i = getname()) != 0) {
762                         if (i == 1) {
```



```
763                                 (void) file_out();
764                                 if (Atflag || SysAtflag) {
765                                         if (Gen.g_dirfd != -1) {
766                                                 (void) close(Gen.g_dirfd);
767                                         }
768                                         Gen.g_dirfd = -1;
769                                         xattrs_out(file_out);
770                                 }
771                         }
772                         if (aclp != NULL) {
773                                 acl_free(aclp);
774                                 aclp = NULL;
775                                 acl_is_set = 0;
776                         }
777                 }
778                 write_trail();
779                 break;
780         case OCp: /* PASS */
781                 sl_preview_synonyms();

783                 Gen.g_dirfd = -1;
784                 Gen.g_passdirfd = -1;
785                 Gen.g_dirpath = NULL;
786                 Compress_sparse = 1;
787                 while (getname()) {
788                         /*
789                          * If file is a fully qualified path then
790                          * file_pass will strip off the leading '/'
791                          * and we need to save off the unstripped
792                          * name for attribute traversal.
793                          */
794                         if (Atflag || SysAtflag) {
795                                 (void) strcpy(Savenam_p, Gen.g_nam_p);
796                         }
797                         passret = file_pass();
798                         if (aclp != NULL) {
799                                 acl_free(aclp);
800                                 aclp = NULL;
801                                 acl_is_set = 0;
802                         }
803                         if (Gen.g_passdirfd != -1)
804                                 (void) close(Gen.g_passdirfd);
805                         Gen.g_passdirfd = -1;
806                         if (Atflag || SysAtflag) {
807                                 if (Gen.g_dirfd != -1) {
808                                         (void) close(Gen.g_dirfd);
809                                 }
810                                 Gen.g_dirfd = -1;
811                                 if (passret != FILE_LINKED) {
812                                         Gen.g_nam_p = Savenam_p;
813                                         xattrs_out(file_pass);
814                                 }
815                         }
816                 }
817                 break;
818         default:
819                 msg(EXT, "Impossible action.");
820         }
821         if (Ofile > 0) {
822                 if (close(Ofile) != 0)
823                         msg(EXTN, "close error");
824         }
825         if (Archive > 0) {
826                 if (close(Archive) != 0)
827                         msg(EXTN, "close error");
828         }
```

```
 829         if ((Args & OCq) == 0) {
 830                 Blocks = (u_longlong_t)(Blocks * Bufsize + SBlocks +
 831                     0x1FF) >> 9;
 828         Blocks = (u_longlong_t)(Blocks * Bufsize + SBlocks + 0x1FF) >> 9;
 832                 msg(EPOST, "%lld blocks", Blocks);
 833         }
 834         if (Error_cnt)
 835                 msg(EPOST, "%d error(s)", Error_cnt);
 836         return (EXIT_CODE);
 837 }
_____unchanged_portion_omitted_

6369 /*
6370  * setup:  Perform setup and initialization functions.  Parse the options
6371  * using getopt(3C), call ckopts to check the options and initialize various
6372  * structures and pointers.  Specifically, for the -i option, save any
6373  * patterns, for the -o option, check (via stat(2)) the archive, and for
6374  * the -p option, validate the destination directory.
6375  */

6377 static void
6378 setup(int largc, char **largv)
6379 {
6380         extern int optind;
6381         extern char *optarg;

6383 #if defined(O_XATTR)
6384 #if defined(_PC_SATTR_ENABLED)
6385 #ifdef WAITAROUND
6386         char    *opts_p = "zabcdfiklmopqrstuvABC:DE:H:I:LM:O:PR:SV6@/";
6382         char    *opts_p = "zabcdfiklmoprstuvABC:DE:H:I:LM:O:PR:SV6@/";
6387 #else
6388         char    *opts_p = "abcdfiklmopqrstuvABC:DE:H:I:LM:O:PR:SV6@/";
6384         char    *opts_p = "abcdfiklmoprstuvABC:DE:H:I:LM:O:PR:SV6@/";
6389 #endif  /* WAITAROUND */

6391 #else   /* _PC_SATTR_ENABLED */
6392 #ifdef WAITAROUND
6393         char    *opts_p = "zabcdfiklmopqrstuvABC:DE:H:I:LM:O:PR:SV6@";
6389         char    *opts_p = "zabcdfiklmoprstuvABC:DE:H:I:LM:O:PR:SV6@";
6394 #else
6395         char    *opts_p = "abcdfiklmopqrstuvABC:DE:H:I:LM:O:PR:SV6@";
6391         char    *opts_p = "abcdfiklmoprstuvABC:DE:H:I:LM:O:PR:SV6@";
6396 #endif  /* WAITAROUND */
6397 #endif  /* _PC_SATTR_ENABLED */

6399 #else   /* O_XATTR */
6400 #ifdef WAITAROUND
6401         char    *opts_p = "zabcdfiklmopqrstuvABC:DE:H:I:LM:O:PR:SV6";
6397         char    *opts_p = "zabcdfiklmoprstuvABC:DE:H:I:LM:O:PR:SV6";
6402 #else
6403         char    *opts_p = "abcdfiklmopqrstuvABC:DE:H:I:LM:O:PR:SV6";
6399         char    *opts_p = "abcdfiklmoprstuvABC:DE:H:I:LM:O:PR:SV6";
6404 #endif  /* WAITAROUND */
6405 #endif  /* O_XATTR */

6407         char   *dupl_p = "Only one occurrence of -%c allowed";
6408         int option;
6409         int blk_cnt, blk_cnt_max;
6410         struct rlimit rlim;

6412         /* Remember the native page size. */

6414         PageSize = sysconf(_SC_PAGESIZE);

6416         if (PageSize == -1) {
```



```
6417                 /*
6418                  * This sysconf call will almost certainly never fail.  The
6419                  * symbol PAGESIZE itself resolves to the above sysconf call,
6420                  * so we should go ahead and define our own constant.
6421                  */
6422                 PageSize = 8192;
6423         }

6425         Hdr_type = BIN;
6426         Max_offset = (off_t)(BIN_OFFSET_MAX);
6427         Efil_p = Hdr_p = Own_p = IOfil_p = NULL;
6428         while ((option = getopt(largc, largv, opts_p)) != EOF) {
6429                 switch (option) {
6430 #ifdef WAITAROUND
6431                 case 'z':
6432                         /* rendezvous with the debugger */
6433                         waitaround = 1;
6434                         break;
6435 #endif
6436                 case 'a':       /* reset access time */
6437                         Args |= OCa;
6438                         break;
6439                 case 'b':       /* swap bytes and halfwords */
6440                         Args |= OCb;
6441                         break;
6442                 case 'c':       /* select character header */
6443                         Args |= OCc;
6444                         Hdr_type = ASC;
6445                         Max_namesz = APATH;
6446                         Onecopy = 1;
6447                         break;
6448                 case 'd':       /* create directories as needed */
6449                         Args |= OCd;
6450                         break;
6451                 case 'f':       /* select files not in patterns */
6452                         Args |= OCf;
6453                         break;
6454                 case 'i':       /* "copy in" */
6455                         Args |= OCi;
6456                         Archive = 0;
6457                         break;
6458                 case 'k':       /* retry after I/O errors */
6459                         Args |= OCk;
6460                         break;
6461                 case 'l':       /* link files when possible */
6462                         Args |= OCl;
6463                         break;
6464                 case 'm':       /* retain modification time */
6465                         Args |= OCm;
6466                         break;
6467                 case 'o':       /* "copy out" */
6468                         Args |= OCo;
6469                         Archive = 1;
6470                         break;
6471                 case 'p':       /* "pass" */
6472                         Max_namesz = APATH;
6473                         Args |= OCp;
6474                         break;
6475                 case 'q':       /* "quiet" */
6476                         Args |= OCq;
6477                         break;
6478                 case 'r':       /* rename files interactively */
6479                         Args |= OCr;
6480                         break;
6481                 case 's':       /* swap bytes */
6482                         Args |= OCs;
```

```
6483                    break;
6484            case 't':       /* table of contents */
6485                    Args |= OCt;
6486                    break;
6487            case 'u':       /* copy unconditionally */
6488                    Args |= OCu;
6489                    break;
6490            case 'v':       /* verbose - print file names */
6491                    Args |= OCv;
6492                    break;
6493            case 'A':       /* append to existing archive */
6494                    Args |= OCA;
6495                    break;
6496            case 'B':       /* set block size to 5120 bytes */
6497                    Args |= OCB;
6498                    Bufsize = 5120;
6499                    break;
6500            case 'C':       /* set arbitrary block size */
6501                    if (Args & OCC)
6502                            msg(ERR, dupl_p, 'C');
6503                    else {
6504                            Args |= OCC;
6505                            Bufsize = atoi(optarg);
6506                    }
6507                    break;
6508            case 'D':
6509                    Dflag = 1;
6510                    break;
6511            case 'E':       /* alternate file for pattern input */
6512                    if (Args & OCE)
6513                            msg(ERR, dupl_p, 'E');
6514                    else {
6515                            Args |= OCE;
6516                            Efil_p = optarg;
6517                    }
6518                    break;
6519            case 'H':       /* select header type */
6520                    if (Args & OCH)
6521                            msg(ERR, dupl_p, 'H');
6522                    else {
6523                            Args |= OCH;
6524                            Hdr_p = optarg;
6525                    }
6526                    break;
6527            case 'I':       /* alternate file for archive input */
6528                    if (Args & OCI)
6529                            msg(ERR, dupl_p, 'I');
6530                    else {
6531                            Args |= OCI;
6532                            IOfil_p = optarg;
6533                    }
6534                    break;
6535            case 'L':       /* follow symbolic links */
6536                    Args |= OCL;
6537                    break;
6538            case 'M':       /* specify new end-of-media message */
6539                    if (Args & OCM)
6540                            msg(ERR, dupl_p, 'M');
6541                    else {
6542                            Args |= OCM;
6543                            Eom_p = optarg;
6544                    }
6545                    break;
6546            case 'O':       /* alternate file for archive output */
6547                    if (Args & OCO)
6548                            msg(ERR, dupl_p, 'O');
```



```
6549                    else {
6550                            Args |= OCO;
6551                            IOfil_p = optarg;
6552                    }
6553                    break;
6554            case 'P':       /* preserve acls */
6555                    Args |= OCP;
6556                    Pflag++;
6557                    break;
6558            case 'R':       /* change owner/group of files */
6559                    if (Args & OCR)
6560                            msg(ERR, dupl_p, 'R');
6561                    else {
6562                            Args |= OCR;
6563                            Own_p = optarg;
6564                    }
6565                    break;
6566            case 'S':       /* swap halfwords */
6567                    Args |= OCS;
6568                    break;
6569            case 'V':       /* print a dot '.' for each file */
6570                    Args |= OCV;
6571                    break;
6572            case '6':       /* for old, sixth-edition files */
6573                    Args |= OC6;
6574                    Ftype = SIXTH;
6575                    break;
6576 #if defined(O_XATTR)
6577            case '@':
6578                    Atflag++;
6579                    break;
6580 #if defined(_PC_SATTR_ENABLED)
6581            case '/':
6582                    SysAtflag++;
6583                    break;
6584 #endif  /* _PC_SATTR_ENABLED */
6585 #endif  /* O_XATTR */
6586            default:
6587                    Error_cnt++;
6588            } /* option */
6589    } /* (option = getopt(largc, largv, opts_p)) != EOF */

6591 #ifdef WAITAROUND
6592    if (waitaround) {
6593            (void) fprintf(stderr, gettext("Rendezvous with cpio on pid"
6594                " %d\n"), getpid());

6596            while (waitaround) {
6597                    (void) sleep(10);
6598            }
6599    }
6600 #endif

6602    largc -= optind;
6603    largv += optind;
6604    ckopts(Args);
6605    if (!Error_cnt) {
6606            if (Args & OCr) {
6607                    Renam_p = e_zalloc(E_EXIT, APATH + 1);
6608                    Renametmp_p = e_zalloc(E_EXIT, APATH + 1);
6609 #if defined(_PC_SATTR_ENABLED)
6610                    Renam_attr_p = e_zalloc(E_EXIT, APATH + 1);
6611 #endif
6612            }
6613            Symlnk_p = e_zalloc(E_EXIT, APATH);
6614            Over_p = e_zalloc(E_EXIT, APATH);
```

```
6615                Nam_p = e_zalloc(E_EXIT, APATH + 1);
6616                if (Args & OCp) {
6617                        Savenam_p = e_zalloc(E_EXIT, APATH + 1);
6618                }
6619                Fullnam_p = e_zalloc(E_EXIT, APATH);
6620                Lnknam_p = e_zalloc(E_EXIT, APATH);
6621                Gen.g_nam_p = Nam_p;
6622                if ((Fullnam_p = getcwd(NULL, APATH)) == NULL)
6623                        msg(EXT, "Unable to determine current directory.");
6624                if (Args & OCi) {
6625                        if (largc > 0) /* save patterns for -i option, if any */
6626                                Pat_pp = largv;
6627                        if (Args & OCE)
6628                                getpats(largc, largv);
6629                } else if (Args & OCo) {
6630                        if (largc != 0) /* error if arguments left with -o */
6631                                Error_cnt++;
6632                        else if (fstat(Archive, &ArchSt) < 0)
6633                                msg(ERRN, "Error during stat() of archive");
6634                        switch (Hdr_type) {
6635                        case BIN:
6636                                Hdrsz = HDRSZ;
6637                                Pad_val = HALFWD;
6638                                break;
6639                        case CHR:
6640                                Hdrsz = CHRSZ;
6641                                Pad_val = 0;
6642                                Max_offset = (off_t)(CHAR_OFFSET_MAX);
6643                                break;
6644                        case ASC:
6645                        case CRC:
6646                                Hdrsz = ASCSZ;
6647                                Pad_val = FULLWD;
6648                                Max_offset = (off_t)(ASC_OFFSET_MAX);
6649                                break;
6650                        case TAR:
6651                        /* FALLTHROUGH */
6652                        case USTAR: /* TAR and USTAR */
6653                                Hdrsz = TARSZ;
6654                                Pad_val = FULLBK;
6655                                Max_offset = (off_t)(CHAR_OFFSET_MAX);
6656                                break;
6657                        default:
6658                                msg(EXT, "Impossible header type.");
6659                        }
6660                } else { /* directory must be specified */
6661                        if (largc != 1)
6662                                Error_cnt++;
6663                        else if (access(*largv, 2) < 0 && (errno != EACCES))
6664                                /*
6665                                 * EACCES is ignored here as it may occur
6666                                 * when any directory component of the path
6667                                 * does not have write permission, even though
6668                                 * the destination subdirectory has write
6669                                 * access. Writing to a read only directory
6670                                 * is handled later, as in "copy in" mode.
6671                                 */
6672                                msg(ERRN,
6673                                    "Error during access() of \"%s\"", *largv);
6674                }
6675        }
6676        if (Error_cnt)
6677                usage(); /* exits! */
6678        if (Args & (OCi | OCo)) {
6679                if (!Dflag) {
6680                        if (Args & (OCB | OCC)) {
```



```
6681                                if (g_init(&Device, &Archive) < 0)
6682                                        msg(EXTN,
6683                                            "Error during initialization");
6684                        } else {
6685                                if ((Bufsize = g_init(&Device, &Archive)) < 0)
6686                                        msg(EXTN,
6687                                            "Error during initialization");
6688                        }
6689                }

6691                blk_cnt_max = _20K / Bufsize;
6692                if (blk_cnt_max < MX_BUFS) {
6693                        blk_cnt_max = MX_BUFS;
6694                }

6696                Buffr.b_base_p = NULL;

6698                for (blk_cnt = blk_cnt_max; blk_cnt > 1; blk_cnt--) {
6699                        Buffr.b_size = (size_t)(Bufsize * blk_cnt);
6700                        Buffr.b_base_p = e_valloc(E_NORMAL, Buffr.b_size);
6701                        if (Buffr.b_base_p != NULL) {
6702                                break;
6703                        }
6704                }
6705                if (Buffr.b_base_p == NULL || Buffr.b_size < (2 * CPIOBSZ)) {
6706                        msg(EXT, "Out of memory");
6707                }

6709                Buffr.b_out_p = Buffr.b_in_p = Buffr.b_base_p;
6710                Buffr.b_cnt = 0L;
6711                Buffr.b_end_p = Buffr.b_base_p + Buffr.b_size;
6712        }

6714        /*
6715         * Now that Bufsize has stabilized, we can allocate our i/o buffer
6716         */
6717        Buf_p = e_valloc(E_EXIT, Bufsize);

6719        if (Args & OCp) { /* get destination directory */
6720                (void) strcpy(Fullnam_p, *largv);
6721                if (stat(Fullnam_p, &DesSt) < 0)
6722                        msg(EXTN, "Error during stat() of \"%s\"", Fullnam_p);
6723                if ((DesSt.st_mode & Ftype) != S_IFDIR)
6724                        msg(EXT, "\"%s\" is not a directory", Fullnam_p);
6725        }
6726        Full_p = Fullnam_p + strlen(Fullnam_p) - 1;
6727        if (*Full_p != '/') {
6728                Full_p++;
6729                *Full_p = '/';
6730        }
6731        Full_p++;
6732        *Full_p = '\0';
6733        (void) strcpy(Lnknam_p, Fullnam_p);
6734        Lnkend_p = Lnknam_p + strlen(Lnknam_p);
6735        (void) getrlimit(RLIMIT_FSIZE, &rlim);
6736        Max_filesz = (off_t)rlim.rlim_cur;
6737        Lnk_hd.L_nxt_p = Lnk_hd.L_bck_p = &Lnk_hd;
6738        Lnk_hd.L_lnk_p = NULL;
6739 }
```

unchanged_portion_omitted_

```
6866 /*
6867  * usage: Print the usage message on stderr and exit.
6868  */

6870 static void
```

```
6871 usage(void)
6872 {

6874         (void) fflush(stdout);
6875 #if defined(O_XATTR)
6876         (void) fprintf(stderr, gettext("USAGE:\n"
6877             "\tcpio -i[bcdfkmqrstuv@BSV6] [-C size] "
6870             "\tcpio -i[bcdfkmrstuv@BSV6] [-C size] "
6878             "[-E file] [-H hdr] [-I file [-M msg]] "
6879             "[-R id] [patterns]\n"
6880             "\tcpio -o[acv@ABLV] [-C size] "
6881             "[-H hdr] [-O file [-M msg]]\n"
6882             "\tcpio -p[adlmuv@LV] [-R id] directory\n"));
6883 #else
6884         (void) fprintf(stderr, gettext("USAGE:\n"
6885             "\tcpio -i[bcdfkmqrstuvBSV6] [-C size] "
6878             "\tcpio -i[bcdfkmrstuvBSV6] [-C size] "
6886             "[-E file] [-H hdr] [-I file [-M msg]] "
6887             "[-R id] [patterns]\n"
6888             "\tcpio -o[acvABLV] [-C size] "
6889             "[-H hdr] [-O file [-M msg]]\n"
6890             "\tcpio -p[adlmuvLV] [-R id] directory\n"));
6891 #endif
6892         (void) fflush(stderr);
6893         exit(EXIT_CODE);
6894 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
    9707 Sun Aug  5 16:35:35 2012
new/usr/src/cmd/cpio/cpio.h
1154 cpio needs a quiet option
**********************************************************
```

```
/*
 * 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) 2012 Gary Mills
 *
 * 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   */

#ifndef _CPIO_H
#define _CPIO_H

#ifdef __cplusplus
extern "C" {
#endif

#include <stdio.h>
#include <archives.h>

/* Option Character keys (OC#), where '#' is the option character specified. */

#define OCa     0x1
#define OCb     0x2
#define OCc     0x4
#define OCd     0x8
#define OCf     0x10
#define OCi     0x20
#define OCk     0x40
#define OCl     0x80
#define OCm     0x100
#define OCo     0x200
#define OCp     0x400
#define OCr     0x800
#define OCs     0x1000
#define OCt     0x2000
#define OCu     0x4000
#define OCv     0x8000
#define OCA     0x10000
#define OCB     0x20000
#define OCC     0x40000
```



```
#define OCE     0x80000
#define OCH     0x100000
#define OCI     0x200000
#define OCL     0x400000
#define OCM     0x800000
#define OCO     0x1000000
#define OCR     0x2000000
#define OCS     0x4000000
#define OCV     0x8000000
#define OC6     0x10000000
#define BSM     0x20000000
#define OCP     0x40000000
#define OCq     0x80000000

/* Sparse file support */
#define C_ISSPARSE      0200000
#define S_IFSPARSE      0x10000
#define HIGH_ORD_MASK   0x30000
#define S_ISSPARSE(mode) \
        (S_ISREG(mode) && (mode & HIGH_ORD_MASK) == S_IFSPARSE)

/* Invalid option masks for each action option (-i, -o or -p). */

#define INV_MSK4i       (OCo | OCp | OCA | OCL | OCO)

#define INV_MSK4o       (OCi | OCp | OCE | OCI | OCR)

#define INV_MSK4p       (OCf | OCi | OCo | OCr | OCt | OCA \
                        | OCE | OCH | OCI | OCO)

/* Header types */

#define NONE    0       /* No header value verified */
#define BIN     1       /* Binary */
#define CHR     2       /* ASCII character (-c) */
#define ASC     3       /* ASCII with expanded maj/min numbers */
#define CRC     4       /* CRC with expanded maj/min numbers */
#define TARTYP  5       /* Tar or USTAR */
#define SECURE  6       /* Secure system */

/* Differentiate between TAR and USTAR */

#define TAR     7       /* Regular tar */
#define USTAR   8       /* IEEE data interchange standard */

#define ULL_MAX_SIZE    20
#define UL_MAX_SIZE     10

/* constants for bar, used for extracting bar archives */
#define BAR     9
#define BAR_VOLUME_MAGIC        'V'
#define BARTYP  7
#define BARSZ   512
#define BAR_TAPE_SIZE   (126*BARSZ)
#define BAR_FLOPPY_SIZE (18*BARSZ)

/* the pathname lengths for the USTAR header */

#define MAXNAM  256     /* The maximum pathname length */
#define NAMSIZ  100     /* The maximum length of the name field */
#define PRESIZ  155     /* The maximum length of the prefix */

/* HDRSZ: header size minus filename field length */

#define HDRSZ (Hdr.h_name - (char *)&Hdr)
```

```
128 /*
129  * IDENT: Determine if two stat() structures represent identical files.
130  * Assumes that if the device and inode are the same the files are
131  * identical (prevents the archive file from appearing in the archive).
132  */

134 #define IDENT(a, b) ((a.st_ino == b.st_ino && a.st_dev == b.st_dev) ? 1 : 0)

136 /*
137  * FLUSH: Determine if enough space remains in the buffer to hold
138  * cnt bytes, if not, call bflush() to flush the buffer to the archive.
139  */

141 #define FLUSH(cnt) if ((Buffr.b_end_p - Buffr.b_in_p) < cnt) bflush()

143 /*
144  * FILL: Determine if enough bytes remain in the buffer to meet current needs,
145  * if not, call rstbuf() to reset and refill the buffer from the archive.
146  */

148 #define FILL(cnt) while (Buffr.b_cnt < cnt) rstbuf()

150 /*
151  * VERBOSE: If x is non-zero, call verbose().
152  */

154 #define VERBOSE(x, name) if (x) verbose(name)

156 /*
157  * FORMAT: Date time formats
158  * b - abbreviated month name
159  * e - day of month (1 - 31)
160  * H - hour (00 - 23)
161  * M - minute (00 - 59)
162  * Y - year as ccyy
163  */

165 #define FORMAT  "%b %e %H:%M %Y"

167 /* Extended system attributes */
168 #ifndef VIEW_READONLY
169 #define VIEW_READONLY   "SUNWattr_ro"
170 #endif

172 #ifndef VIEW_READWRITE
173 #define VIEW_READWRITE  "SUNWattr_rw"
174 #endif

176 #define min(a, b)       ((a) < (b) ? (a) : (b))

178 /* Values used in typeflag field */
179 #define REGTYPE         '0'             /* Regular File */
180 #define LNKTYPE         '1'             /* Link */
181 #define SYMTYPE         '2'             /* Reserved */
182 #define CHRTYPE         '3'             /* Character Special File */
183 #define BLKTYPE         '4'             /* Block Special File */
184 #define DIRTYPE         '5'             /* Directory */
185 #define FIFOTYPE        '6'             /* FIFO */
186 #define CONTTYPE        '7'             /* Reserved */
187 #define XHDRTYPE        'X'             /* Extended header */

189 #define INPUT   0       /* -i mode (used for chgreel() */
190 #define OUTPUT  1       /* -o mode (used for chgreel() */
191 #define APATH   1024    /* maximum ASC or CRC header path length */
192 #define CPATH   256     /* maximum -c and binary path length */
193 #define BUFSZ   512     /* default buffer size for archive I/O */
```



```
194 #define CPIOBSZ 8192    /* buffer size for file system I/O */
195 #define LNK_INC 500     /* link allocation increment */
196 #define MX_BUFS 10      /* max. number of buffers to allocate */

198 #define F_SKIP  0       /* an object did not match the patterns */
199 #define F_LINK  1       /* linked file */
200 #define F_EXTR  2       /* extract non-linked object that matched patterns */

202 #define MX_SEEKS        10      /* max. number of lseek attempts after error */
203 #define SEEK_ABS        0       /* lseek absolute */
204 #define SEEK_REL        1       /* lseek relative */

206 /*
207  * xxx_CNT represents the number of sscanf items that will be matched
208  * if the sscanf to read a header is successful.  If sscanf returns a number
209  * that is not equal to this, an error occured (which indicates that this
210  * is not a valid header of the type assumed.
211  */

213 #define ASC_CNT 14      /* ASC and CRC headers */
214 #define CHR_CNT 11      /* CHR header */

216 /* These defines determine the severity of the message sent to the user. */

218 #define ERR     1       /* Error message (warning) - not fatal */
219 #define EXT     2       /* Error message - fatal, causes exit */
220 #define ERRN    3       /* Error message with errno (warning) - not fatal */
221 #define EXTN    4       /* Error message with errno - fatal, causes exit */
222 #define POST    5       /* Information message, not an error */
223 #define EPOST   6       /* Information message to stderr */

225 #define SIXTH   060000  /* UNIX 6th edition files */

227 #define P_SKIP  0       /* File should be skipped */
228 #define P_PROC  1       /* File should be processed */

230 #define U_KEEP  0       /* Keep the existing version of a file (-u) */
231 #define U_OVER  1       /* Overwrite the existing version of a file (-u) */

233 /*
234  * _20K: Allocate the maximum of (20K or (MX_BUFS * Bufsize)) bytes
235  * for the main I/O buffer.  Therefore if a user specifies a small buffer
236  * size, they still get decent performance due to the buffering strategy.
237  */

239 #define _20K    20480

241 #define HALFWD  1       /* Pad headers/data to halfword boundaries */
242 #define FULLWD  3       /* Pad headers/data to word boundaries */
243 #define FULLBK  511     /* Pad headers/data to 512 byte boundaries */

245 /* bar structure */
246 union b_block {
247         char dummy[TBLOCK];
248         struct bar_header {
249                 char mode[8];
250                 char uid[8];
251                 char gid[8];
252                 char size[12];
253                 char mtime[12];
254                 char chksum[8];
255                 char rdev[8];
256                 char linkflag;

258                 /*
259                  * The following fields are specific to the volume
```

```
 260                  * header.  They are set to zero in all file headers
 261                  * in the archive.
 262                  */
 263                 char bar_magic[2];      /* magic number */
 264                 char volume_num[4];     /* volume number */
 265                 char compressed;        /* files compressed = 1 */
 266                 char date[12];          /* date of archive mmddhhmm */
 267                 char start_of_name;     /* start of the filename */
 268         } dbuf;
 269 };
```

unchanged_portion_omitted_

```
**********************************************************
   25659 Sun Aug  5 16:35:35 2012
new/usr/src/man/man1/cpio.1
1154 cpio needs a quiet option
**********************************************************
```

```
   1 '\" te
   2 .\" Copyright 1989 AT&T
   3 .\" Copyright (c) 2009, Sun Microsystems, Inc. All Rights Reserved
   4 .\" Copyright (c) 2012 Gary Mills
   5 .\" Portions Copyright (c) 1992, X/Open Company Limited All Rights Reserved
   6 .\" Sun Microsystems, Inc. gratefully acknowledges The Open Group for permission
   7 .\" The Institute of Electrical and Electronics Engineers and The Open Group, ha
   8 .\" are reprinted and reproduced in electronic form in the Sun OS Reference Manu
   9 .\" and Electronics Engineers, Inc and The Open Group. In the event of any discr
  10 .\"  This notice shall appear on any product containing this material.
  11 .\" The contents of this file are subject to the terms of the Common Development
  12 .\"  See the License for the specific language governing permissions and limitat
  13 .\" the fields enclosed by brackets "[]" replaced with your own identifying info
  14 .TH CPIO 1 "Aug 3, 2009"
  15 .SH NAME
  16 cpio \- copy file archives in and out
  17 .SH SYNOPSIS
  18 .LP
  19 .nf
  20 \fBcpio\fR \fB-i\fR [\fB-bBcdfkmPqrsStuvV6@/\fR] [\fB-C\fR \fIbufsize\fR] [\fB-E
  19 \fBcpio\fR \fB-i\fR [\fB-bBcdfkmPrsStuvV6@/\fR] [\fB-C\fR \fIbufsize\fR] [\fB-E\
  21      [\fB-H\fR \fIheader\fR] [\fB-I\fR \fI\fR [\fB-M\fR \fImessage\fR]] [\fB-R\f
  22 .fi

  24 .LP
  25 .nf
  26 \fBcpio\fR \fB-o\fR [\fB-aABcLPqvV@/\fR] [\fB-C\fR \fIbufsize\fR] [\fB-H\fR \fIh
  25 \fBcpio\fR \fB-o\fR [\fB-aABcLPvV@/\fR] [\fB-C\fR \fIbufsize\fR] [\fB-H\fR \fIhe
  27      [\fB-O\fR \fIfile\fR [\fB-M\fR \fImessage\fR]]
  28 .fi

  30 .LP
  31 .nf
  32 \fBcpio\fR \fB-p\fR [\fB-adlLmPquvV@/\fR] [\fB-R\fR \fIid\fR] \fIdirectory\fR
  31 \fBcpio\fR \fB-p\fR [\fB-adlLmPuvV@/\fR] [\fB-R\fR \fIid\fR] \fIdirectory\fR
  33 .fi

  35 .SH DESCRIPTION
  36 .sp
  37 .LP
  38 The \fBcpio\fR command copies files into and out of a \fBcpio\fR archive. The
  39 \fBcpio\fR archive can span multiple volumes. The \fB-i\fR, \fB-o\fR, and
  40 \fB-p\fR options select the action to be performed. The following list
  41 describes each of the actions. These actions are mutually exclusive.
  42 .SS "Copy In Mode"
  43 .sp
  44 .LP
  45 \fBcpio\fR \fB-i\fR (copy in) extracts files from the standard input, which is
  46 assumed to be the product of a previous \fBcpio\fR \fB-o\fR command. Only files
  47 with names that match one of the \fIpattern\fRs are selected. See \fBsh\fR(1)
  48 and OPERANDS for more information about \fIpattern\fR. Extracted files are
  49 conditionally copied into the current directory tree, based on the options
  50 described below. The permissions of the files are those of the previous \fBcpio
  51 -o\fR command. The owner and group are the same as the current user, unless the
  52 current user has the \fB{PRIV_FILE_CHOWN_SELF}\fR privilege. See
  53 \fBchown\fR(2). If this is the case, owner and group are the same as those
  54 resulting from the previous \fBcpio -o\fR command. Notice that if \fBcpio\fR
  55 \fB-i\fR tries to create a file that already exists and the existing file is
  56 the same age or younger (\fBnewer\fR), \fBcpio\fR outputs a warning message and
  57 not replace the file. The \fB-u\fR option can be used to unconditionally
  58 overwrite the existing file.
```



```
  59 .SS "Copy Out Mode"
  60 .sp
  61 .LP
  62 \fBcpio\fR \fB-o\fR (copy out) reads a list of file path names from the
  63 standard input and copies those files to the standard output, together with
  64 path name and status information in the form of a \fBcpio\fR archive. Output is
  65 padded to an 8192-byte boundary by default or to the user-specified block size
  66 (with the \fB-B\fR or \fB-C\fR options) or to some device-dependent block size
  67 where necessary (as with the CTC tape).
  68 .SS "Pass Mode"
  69 .sp
  70 .LP
  71 \fBcpio\fR \fB-p\fR (pass) reads a list of file path names from the standard
  72 input and conditionally copies those files into the destination directory tree,
  73 based on the options described below.
  74 .sp
  75 .LP
  76 If the underlying file system of the source file supports detection of holes as
  77 reported by \fBpathconf\fR(2), the file is a sparse file, and the destination
  78 file is seekable, then holes in sparse files are preserved in pass mode,
  79 otherwise holes are filled with zeros.
  80 .sp
  81 .LP
  82 \fBcpio\fR assumes four-byte words.
  83 .sp
  84 .LP
  85 If, when writing to a character device (\fB-o\fR) or reading from a character
  86 device (\fB-i\fR), \fBcpio\fR reaches the end of a medium (such as the end of a
  87 diskette), and the \fB-O\fR and \fB-I\fR options are not used, \fBcpio\fR
  88 prints the following message:
  89 .sp
  90 .in +2
  91 .nf
  92 To continue, type device/file name when ready.
  93 .fi
  94 .in -2
  95 .sp

  97 .sp
  98 .LP
  99 To continue, you must replace the medium and type the character special device
 100 name (\fB/dev/rdiskette\fR for example) and press RETURN. You might want to
 101 continue by directing \fBcpio\fR to use a different device. For example, if you
 102 have two floppy drives you might want to switch between them so \fBcpio\fR can
 103 proceed while you are changing the floppies. Press RETURN to cause the
 104 \fBcpio\fR process to exit.
 105 .SH OPTIONS
 106 .sp
 107 .LP
 108 The following options are supported:
 109 .sp
 110 .ne 2
 111 .na
 112 \fB\fB-i\fR\fR
 113 .ad
 114 .RS 6n
 115 (copy in) Reads an archive from the standard input and conditionally extracts
 116 the files contained in it and places them into the current directory tree.
 117 .RE

 119 .sp
 120 .ne 2
 121 .na
 122 \fB\fB-o\fR\fR
 123 .ad
 124 .RS 6n
```

(copy out) Reads a list of file path names from the standard input and copies those files to the standard output in the form of a \fBcpio\fR archive.
.RE

.sp
.ne 2
.na
\fB\fB-p\fR\fR
.ad
.RS 6n
(pass) Reads a list of file path names from the standard input and conditionally copies those files into the destination directory tree.
.RE

.sp
.LP
The following options can be appended in any sequence to the \fB-i\fR, \fB-o\fR, or \fB-p\fR options:
.sp
.ne 2
.na
\fB\fB-a\fR\fR
.ad
.RS 14n
Resets access times of input files after they have been copied, making \fBcpio\fR's access invisible. Access times are not reset for linked files when \fBcpio\fR \fB-pla\fR is specified.
.RE

.sp
.ne 2
.na
\fB\fB-A\fR\fR
.ad
.RS 14n
Appends files to an archive. The \fB-A\fR option requires the \fB-O\fR option. Valid only with archives that are files, or that are on floppy diskettes or hard disk partitions. The effect on files that are linked in the existing portion of the archive is unpredictable.
.RE

.sp
.ne 2
.na
\fB\fB-b\fR\fR
.ad
.RS 14n
Reverses the order of the bytes within each word. Use only with the \fB-i\fR option.
.RE

.sp
.ne 2
.na
\fB\fB-B\fR\fR
.ad
.RS 14n
Blocks input/output 5120 bytes to the record. The default buffer size is 8192 bytes when this and the \fB-C\fR options are not used. \fB-B\fR does not apply to the \fB-p\fR (pass) option.
.RE

.sp
.ne 2
.na
\fB\fB-c\fR\fR

.ad
.RS 14n
Reads or writes header information in \fBASCII\fR character form for portability. There are no \fBUID\fR or \fBGID\fR restrictions associated with this header format. Use this option between SVR4-based machines, or the \fB-H\fR \fBodc\fR option between unknown machines. The \fB-c\fR option implies the use of expanded device numbers, which are only supported on SVR4-based systems. When transferring files between SunOS 4 or Interactive UNIX and the Solaris 2.6 Operating environment or compatible versions, use \fB-H\fR \fBodc\fR.
.RE

.sp
.ne 2
.na
\fB\fB-C\fR \fIbufsize\fR\fR
.ad
.RS 14n
Blocks input/output \fIbufsize\fR bytes to the record, where \fIbufsize\fR is replaced by a positive integer. The default buffer size is 8192 bytes when this and \fB-B\fR options are not used. \fB-C\fR does not apply to the \fB-p\fR (pass) option.
.RE

.sp
.ne 2
.na
\fB\fB-d\fR\fR
.ad
.RS 14n
Creates directories as needed.
.RE

.sp
.ne 2
.na
\fB\fB-E\fR \fIfile\fR\fR
.ad
.RS 14n
Specifies an input file (\fIfile\fR) that contains a list of filenames to be extracted from the archive (one filename per line).
.RE

.sp
.ne 2
.na
\fB\fB-f\fR\fR
.ad
.RS 14n
Copies in all files except those in \fIpattern\fRs. See OPERANDS for a description of \fIpattern\fR.
.RE

.sp
.ne 2
.na
\fB\fB-H\fR \fIheader\fR\fR
.ad
.RS 14n
Reads or writes header information in \fIheader\fR format. Always use this option or the \fB-c\fR option when the origin and the destination machines are different types. This option is mutually exclusive with options \fB-c\fR and \fB-6\fR.
.sp
Valid values for \fIheader\fR are:
.sp

```
.ne 2
.na
\fB\fBbar\fR\fR
.ad
.RS 17n
\fBbar\fR head and format. Used only with the \fB-i\fR option ( read only).
.RE

.sp
.ne 2
.na
\fB\fBcrc\fR | \fBCRC\fR\fR
.ad
.RS 17n
\fBASCII\fR header with expanded device numbers and an additional per-file
checksum. There are no \fBUID\fR or \fBGID\fR restrictions associated with this
header format.
.RE

.sp
.ne 2
.na
\fB\fBodc\fR\fR
.ad
.RS 17n
\fBASCII\fR header with small device numbers. This is the IEEE/P1003 Data
Interchange Standard cpio header and format. It has the widest range of
portability of any of the header formats. It is the official format for
transferring files between POSIX-conforming systems (see \fBstandards\fR(5)).
Use this format to communicate with SunOS 4 and Interactive UNIX. This header
format allows \fBUID\fRs and \fBGID\fRs up to 262143 to be stored in the
header.
.RE

.sp
.ne 2
.na
\fB\fBtar\fR | \fBTAR\fR\fR
.ad
.RS 17n
\fBtar\fR header and format. This is an older \fBtar\fR header format that
allows \fBUID\fRs and \fBGID\fRs up to 2097151 to be stored in the header. It
is provided for the reading of legacy archives only, that is, in conjunction
with option \fB-i\fR.
.sp
Specifying this archive format with option \fB-o\fR has the same effect as
specifying the "ustar" format: the output archive is in \fBustar\fR format, and
must be read using \fB-H\fR \fBustar\fR.
.RE

.sp
.ne 2
.na
\fB\fBustar\fR | \fBUSTAR\fR\fR
.ad
.RS 17n
IEEE/P1003 Data Interchange Standard tar header and format. This header format
allows \fBUID\fRs and \fBGID\fRs up to 2097151 to be stored in the header.
.RE

Files with \fBUID\fRs and \fBGID\fRs greater than the limit stated above are
archived with the \fBUID\fR and \fBGID\fR of \fB60001\fR. To transfer a large
file (8 Gb \(em 1 byte), the header format can be \fBtar|TAR\fR,
\fBustar|USTAR\fR, or \fBodc\fR only.
.RE
```



```
.sp
.ne 2
.na
\fB\fB-I\fR \fIfile\fR\fR
.ad
.RS 14n
Reads the contents of \fIfile\fR as an input archive, instead of the standard
input. If \fIfile\fR is a character special device, and the current medium has
been completely read, replace the medium and press RETURN to continue to the
next medium. This option is used only with the \fB-i\fR option.
.RE

.sp
.ne 2
.na
\fB\fB-k\fR\fR
.ad
.RS 14n
Attempts to skip corrupted file headers and I/O errors that might be
encountered. If you want to copy files from a medium that is corrupted or out
of sequence, this option lets you read only those files with good headers. For
\fBcpio\fR archives that contain other \fBcpio\fR archives, if an error is
encountered, \fBcpio\fR can terminate prematurely. \fBcpio\fR finds the next
good header, which can be one for a smaller archive, and terminate when the
smaller archive’s trailer is encountered. Use only with the \fB-i\fR option.
.RE

.sp
.ne 2
.na
\fB\fB-l\fR\fR
.ad
.RS 14n
In pass mode, makes hard links between the source and destination whenever
possible. If the \fB-L\fR option is also specified, the hard link is to the
file referred to by the symbolic link. Otherwise, the hard link is to the
symbolic link itself. Use only with the \fB-p\fR option.
.RE

.sp
.ne 2
.na
\fB\fB-L\fR\fR
.ad
.RS 14n
Follows symbolic links. If a symbolic link to a directory is encountered,
archives the directory referred to by the link, using the name of the link.
Otherwise, archives the file referred to by the link, using the name of the
link.
.RE

.sp
.ne 2
.na
\fB\fB-m\fR\fR
.ad
.RS 14n
Retains previous file modification time. This option is ineffective on
directories that are being copied.
.RE

.sp
.ne 2
.na
\fB\fB-M\fR \fImessage\fR\fR
.ad
```

```
.RS 14n
Defines a \fImessage\fR to use when switching media. When you use the \fB-O\fR
or \fB-I\fR options and specify a character special device, you can use this
option to define the message that is printed when you reach the end of the
medium. One \fB%d\fR can be placed in \fImessage\fR to print the sequence
number of the next medium needed to continue.
.RE

.sp
.ne 2
.na
\fB\fB-O\fR \fIfile\fR\fR
.ad
.RS 14n
Directs the output of \fBcpio\fR to \fIfile\fR, instead of the standard output.
If \fIfile\fR is a character special device and the current medium is full,
replace the medium and type a carriage return to continue to the next medium.
Use only with the \fB-o\fR option.
.RE

.sp
.ne 2
.na
\fB\fB-P\fR\fR
.ad
.RS 14n
Preserves \fBACL\fRs. If the option is used for output, existing \fBACL\fRs are
written along with other attributes, except for extended attributes, to the
standard output. \fBACL\fRs are created as special files with a special file
type. If the option is used for input, existing \fBACL\fRs are extracted along
with other attributes from standard input. The option recognizes the special
file type. Notice that errors occurs if a \fBcpio\fR archive with \fBACL\fRs is
extracted by previous versions of \fBcpio\fR. This option should not be used
with the \fB-c\fR option, as \fBACL\fR support might not be present on all
systems, and hence is not portable. Use \fBASCII\fR headers for portability.
.RE

.sp
.ne 2
.na
\fB\fB-q\fR\fR
.ad
.RS 14n
Quiet. Suppresses the number of blocks message that normally is printed
after the copy is completed.
.RE

.sp
.ne 2
.na
\fB\fB-r\fR\fR
.ad
.RS 14n
Interactively renames files. If the user types a carriage return alone, the
file is skipped. If the user types a ``.'', the original pathname is retained.
Not available with \fBcpio\fR \fB-p\fR.
.RE

.sp
.ne 2
.na
\fB\fB-R\fR \fIid\fR\fR
.ad
.RS 14n
Reassigns ownership and group information for each file to user ID. (ID must be
a valid login ID from the \fBpasswd\fR database.) This option is valid only
```



```
when id is the invoking user or the super-user. See \fBNOTES\fR.
.RE

.sp
.ne 2
.na
\fB\fB-s\fR\fR
.ad
.RS 14n
Swaps bytes within each half word.
.RE

.sp
.ne 2
.na
\fB\fB-S\fR\fR
.ad
.RS 14n
Swaps halfwords within each word.
.RE

.sp
.ne 2
.na
\fB\fB-t\fR\fR
.ad
.RS 14n
Prints a table of contents of the input. If any file in the table of contents
has extended attributes, these are also listed. No files are created. \fB-t\fR
and \fB-V\fR are mutually exclusive.
.RE

.sp
.ne 2
.na
\fB\fB-u\fR\fR
.ad
.RS 14n
Copies unconditionally. Normally, an older file is not replaced a newer file
with the same name, although an older directory updates a newer directory.
.RE

.sp
.ne 2
.na
\fB\fB-v\fR\fR
.ad
.RS 14n
Verbose. Prints a list of file and extended attribute names. When used with the
\fB-t\fR option, the table of contents looks like the output of an \fBls\fR
\fB-l\fR command (see \fBls\fR(1)).
.RE

.sp
.ne 2
.na
\fB\fB-V\fR\fR
.ad
.RS 14n
Special verbose. Prints a dot for each file read or written. Useful to assure
the user that \fBcpio\fR is working without printing out all file names.
.RE

.sp
.ne 2
.na
```

```
521 \fB\fB-6\fR\fR
522 .ad
523 .RS 14n
524 Processes a UNIX System Sixth Edition archive format file. Use only with the
525 \fB-i\fR option. This option is mutually exclusive with \fB-c\fR and \fB-H\fR.
526 .RE

528 .sp
529 .ne 2
530 .na
531 \fB\fB-@\fR\fR
532 .ad
533 .RS 14n
534 Includes extended attributes in archive. By default, \fBcpio\fR does not place
535 extended attributes in the archive. With this flag, \fBcpio\fR looks for
536 extended attributes on the files to be placed in the archive and add them, as
537 regular files, to the archive. The extended attribute files go in the archive
538 as special files with special file types. When the \fB-@\fR flag is used with
539 \fB-i\fR or \fB-p\fR, it instructs \fBcpio\fR to restore extended attribute
540 data along with the normal file data. Extended attribute files can only be
541 extracted from an archive as part of a normal file extract. Attempts to
542 explicitly extract attribute records are ignored.
543 .RE

545 .sp
546 .ne 2
547 .na
548 \fB\fB-/\fR\fR
549 .ad
550 .RS 14n
551 Includes extended system attributes in archive. By default, \fBcpio\fR does not
552 place extended system attributes in the archive. With this flag, \fBcpio\fR
553 looks for extended system attributes on the files to be placed in the archive
554 and add them, as regular files, to the archive. The extended attribute files go
555 in the archive as special files with special file types. When the \fB-/\fR flag
556 is used with \fB-i\fR or \fB-p\fR, it instructs \fBcpio\fR to restore extended
557 system attribute data along with the normal file data. Extended system
558 attribute files can only be extracted from an archive as part of a normal file
559 extract. Attempts to explicitly extract attribute records are ignored.
560 .RE

562 .SH OPERANDS
563 .sp
564 .LP
565 The following operands are supported:
566 .sp
567 .ne 2
568 .na
569 \fB\fIdirectory\fR\fR
570 .ad
571 .RS 13n
572 A path name of an existing directory to be used as the target of \fBcpio\fR
573 \fB-p\fR.
574 .RE

576 .sp
577 .ne 2
578 .na
579 \fB\fIpattern\fR\fR
580 .ad
581 .RS 13n
582 Expressions making use of a pattern-matching notation similar to that used by
583 the shell (see \fBsh\fR(1)) for filename pattern matching, and similar to
584 regular expressions. The following metacharacters are defined:
585 .sp
586 .ne 2
```



```
587 .na
588 \fB\fB*\fR\fR
589 .ad
590 .RS 9n
591 Matches any string, including the empty string.
592 .RE

594 .sp
595 .ne 2
596 .na
597 \fB\fB?\fR\fR
598 .ad
599 .RS 9n
600 Matches any single character.
601 .RE

603 .sp
604 .ne 2
605 .na
606 \fB\fB[...]\fR\fR
607 .ad
608 .RS 9n
609 Matches any one of the enclosed characters. A pair of characters separated by
610 '\(mi' matches any symbol between the pair (inclusive), as defined by the
611 system default collating sequence. If the first character following the opening
612 \fB'['\fR is a \fB'!'\fR, the results are unspecified.
613 .RE

615 .sp
616 .ne 2
617 .na
618 \fB\fB!\fR\fR
619 .ad
620 .RS 9n
621 The ! (exclamation point) means \fInot\fR. For example, the \fB!abc*\fR pattern
622 would exclude all files that begin with \fBabc\fR.
623 .RE

625 In \fIpattern\fR, metacharacters \fB?\fR, \fB*\fR, and \fB[\fR\|.\|.\|.\fB]\fR
626 match the slash (\fB/\fR) character, and backslash (\fB\e\fR) is an escape
627 character. Multiple cases of \fIpattern\fR can be specified and if no
628 \fIpattern\fR is specified, the default for \fIpattern\fR is \fB*\fR (that is,
629 select all files).
630 .sp
631 Each pattern must be enclosed in double quotes. Otherwise, the name of a file
632 in the current directory might be used.
633 .RE

635 .SH USAGE
636 .sp
637 .LP
638 See \fBlargefile\fR(5) for the description of the behavior of \fBcpio\fR when
639 encountering files greater than or equal to 2 Gbyte ( 2^31 bytes).
640 .SH EXAMPLES
641 .sp
642 .LP
643 The following examples show three uses of \fBcpio\fR.
644 .LP
645 \fBExample 1 \fRUsing standard input
646 .sp
647 .in +2
648 .nf
649 example% \fBls | cpio -oc > ../newfile\fR
650 .fi
651 .in -2
652 .sp
```

```
654 .sp
655 .LP
656 When standard input is directed through a pipe to \fBcpio\fR \fB-o\fR, as in
657 the example above, it groups the files so they can be directed (>) to a single
658 file (\fB\&../newfile\fR). The \fB-c\fR option insures that the file is
659 portable to other machines (as would the \fB-H\fR option). Instead of
660 \fBls\fR(1), you could use \fBfind\fR(1), \fBecho\fR(1), \fBcat\fR(1), and so
661 on, to pipe a list of names to \fBcpio\fR. You could direct the output to a
662 device instead of a file.

664 .LP
665 \fBExample 2 \fRExtracting files into directories
666 .sp
667 .in +2
668 .nf
669 example% \fBcat newfile | cpio -icd "memo/a1" "memo/b*"\fR
670 .fi
671 .in -2
672 .sp

674 .sp
675 .LP
676 In this example, \fBcpio\fR \fB-i\fR uses the output file of \fBcpio\fR
677 \fB-o\fR (directed through a pipe with \fBcat\fR), extracts those files that
678 match the patterns (\fBmemo/a1\fR, \fBmemo/b*\fR), creates directories below
679 the current directory as needed (\fB-d\fR option), and places the files in the
680 appropriate directories. The \fB-c\fR option is used if the input file was
681 created with a portable header. If no patterns were given, all files from
682 \fBnewfile\fR would be placed in the directory.

684 .LP
685 \fBExample 3 \fRCopying or linking files to another directory
686 .sp
687 .in +2
688 .nf
689 example% \fBfind . -depth -print | cpio -pdlmv newdir\fR
690 .fi
691 .in -2
692 .sp

694 .sp
695 .LP
696 In this example, \fBcpio\fR \fB-p\fR takes the file names piped to it and
697 copies or links (\fB-l\fR option) those files to another directory,
698 \fBnewdir\fR. The \fB-d\fR option says to create directories as needed. The
699 \fB-m\fR option says to retain the modification time. (It is important to use
700 the \fB-depth\fR option of \fBfind\fR(1) to generate path names for \fBcpio\fR.
701 This eliminates problems that \fBcpio\fR could have trying to create files
702 under read-only directories.) The destination directory, \fBnewdir\fR, must
703 exist.

705 .sp
706 .LP
707 Notice that when you use \fBcpio\fR in conjunction with \fBfind\fR, if you use
708 the \fB-L\fR option with \fBcpio\fR, you must use the \fB-follow\fR option with
709 \fBfind\fR and vice versa. Otherwise, there are undesirable results.
710 .sp
711 .LP
712 For multi-reel archives, dismount the old volume, mount the new one, and
713 continue to the next tape by typing the name of the next device (probably the
714 same as the first reel). To stop, type a RETURN and \fBcpio\fR ends.
715 .SH ENVIRONMENT VARIABLES
716 .sp
717 .LP
718 See \fBenviron\fR(5) for descriptions of the following environment variables
```



```
719 that affect the execution of \fBcpio\fR: \fBLC_COLLATE\fR, \fBLC_CTYPE\fR,
720 \fBLC_MESSAGES\fR, \fBLC_TIME\fR, \fBTZ\fR, and \fBNLSPATH\fR.
721 .sp
722 .ne 2
723 .na
724 \fB\fBTMPDIR\fR\fR
725 .ad
726 .RS 10n
727 \fBcpio\fR creates its temporary file in \fB/var/tmp\fR by default. Otherwise,
728 it uses the directory specified by \fBTMPDIR\fR.
729 .RE

731 .SH EXIT STATUS
732 .sp
733 .LP
734 The following exit values are returned:
735 .sp
736 .ne 2
737 .na
738 \fB\fB0\fR\fR
739 .ad
740 .RS 6n
741 Successful completion.
742 .RE

744 .sp
745 .ne 2
746 .na
747 \fB\fB>0\fR\fR
748 .ad
749 .RS 6n
750 An error occurred.
751 .RE

753 .SH ATTRIBUTES
754 .sp
755 .LP
756 See \fBattributes\fR(5) for descriptions of the following attributes:
757 .sp

759 .sp
760 .TS
761 box;
762 c | c
763 l | l .
764 ATTRIBUTE TYPE  ATTRIBUTE VALUE
765 _
766 CSI     Enabled
767 _
768 Interface Stability     Committed
769 .TE

771 .SH SEE ALSO
772 .sp
773 .LP
774 \fBar\fR(1), \fBcat\fR(1), \fBecho\fR(1), \fBfind\fR(1), \fBls\fR(1),
775 \fBpax\fR(1), \fBsetfacl\fR(1), \fBsh\fR(1), \fBtar\fR(1), \fBchown\fR(2),
776 \fBarchives.h\fR(3HEAD), \fBattributes\fR(5), \fBenviron\fR(5),
777 \fBfsattr\fR(5), \fBlargefile\fR(5), \fBstandards\fR(5)
778 .SH NOTES
779 .sp
780 .LP
781 The maximum path name length allowed in a \fBcpio\fR archive is determined by
782 the header type involved. The following table shows the proper value for each
783 supported archive header type.
784 .sp
```

```
.sp
.TS
c c c
l l l .
Header type     Command line options    Maximum path name length
BINARY  "\fB-o\fR"      256
POSIX   "\fB-oH\fR odc" 256
ASCII   "\fB-oc\fR"     1023
CRC     "\fB-oH\fR crc" 1023
USTAR   "\fB-oH\fR ustar"       255
.TE

.sp
.LP
When the command line options "\fB-o\fR \fB-H\fR \fBtar\fR" are specified, the
archive created is of type \fBUSTAR\fR. This means that it is an error to read
this same archive using the command line options "\fB-i\fR \fB-H\fR \fBtar\fR".
The archive should be read using the command line options "\fB-i\fR \fB-H\fR
\fBustar\fR". The options "\fB-i\fR \fB-H\fR \fBtar\fR" refer to an older tar
archive format.
.sp
.LP
An error message is output for files whose \fBUID\fR or \fBGID\fR are too large
to fit in the selected header format. Use \fB-H\fR \fBcrc\fR or \fB-c\fR to
create archives that allow all \fBUID\fR or \fBGID\fR values.
.sp
.LP
Only the super-user can copy special files.
.sp
.LP
Blocks are reported in 512-byte quantities.
.sp
.LP
If a file has \fB000\fR permissions, contains more than 0 characters of data,
and the user is not root, the file is not saved or restored.
.sp
.LP
When cpio is invoked in \fBCopy In\fR or \fBPass Mode\fR by a user with
\fB{PRIV_FILE_CHOWN_SELF}\fR privilege, and in particular on a system where
\fB{_POSIX_CHOWN_RESTRICTED}\fR is not in effect (effectively granting this
privilege to all users where not overridden), extracted or copied files can end
up with owners and groups determined by those of the original archived files,
which can differ from the invoking user's. This might not be what the user
intended. The \fB-R\fR option can be used to retain file ownership, if desired,
if you specify the user's id.
.sp
.LP
The inode number stored in the header (\fB/usr/include/archives.h\fR) is an
unsigned short, which is 2 bytes. This limits the range of inode numbers from
\fB0\fR to \fB65535\fR. Files which are hard linked must fall in this inode
range. This could be a problem when moving \fBcpio\fR archives between
different vendors' machines.
.sp
.LP
You must use the same blocking factor when you retrieve or copy files from the
tape to the hard disk as you did when you copied files from the hard disk to
the tape. Therefore, you must specify the \fB-B\fR or \fB-C\fR option.
.sp
.LP
During \fB-p\fR and \fB-o\fR processing, \fBcpio\fR buffers the file list
presented on stdin in a temporary file.
.sp
.LP
The new \fBpax\fR(1) format, with a command that supports it (for example,
\fBtar\fR), should be used for large files. The \fBcpio\fR command is no longer
```



```
part of the current POSIX standard and is deprecated in favor of \fBpax\fR.
```