```
**********************************************************
  237703 Wed Jan 16 16:49:49 2013
new/usr/src/cmd/tar/tar.c
3474 tar should support -C on extract
Reviewed by: Robert Mustacchi <rm@joyent.com>
**********************************************************
```

```
   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) 2013, Joyent, Inc. All rights reserved.
  25  */

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

  30 /*      Copyright (c) 1987, 1988 Microsoft Corporation  */
  31 /*        All Rights Reserved   */

  33 /*
  34  * Portions of this source code were derived from Berkeley 4.3 BSD
  35  * under license from the Regents of the University of California.
  36  */

  38 #include <unistd.h>
  39 #include <sys/types.h>
  40 #include <sys/param.h>
  41 #include <sys/stat.h>
  42 #include <sys/mkdev.h>
  43 #include <sys/wait.h>
  44 #include <dirent.h>
  45 #include <errno.h>
  46 #include <stdio.h>
  47 #include <signal.h>
  48 #include <ctype.h>
  49 #include <locale.h>
  50 #include <nl_types.h>
  51 #include <langinfo.h>
  52 #include <pwd.h>
  53 #include <grp.h>
  54 #include <fcntl.h>
  55 #include <string.h>
  56 #include <malloc.h>
  57 #include <time.h>
  58 #include <utime.h>
  59 #include <stdlib.h>
  60 #include <stdarg.h>
```



```
  61 #include <widec.h>
  62 #include <sys/mtio.h>
  63 #include <sys/acl.h>
  64 #include <strings.h>
  65 #include <deflt.h>
  66 #include <limits.h>
  67 #include <iconv.h>
  68 #include <assert.h>
  69 #include <libgen.h>
  70 #include <libintl.h>
  71 #include <aclutils.h>
  72 #include <libnvpair.h>
  73 #include <archives.h>

  75 #if defined(__SunOS_5_6) || defined(__SunOS_5_7)
  76 extern int defcntl();
  77 #endif
  78 #if defined(_PC_SATTR_ENABLED)
  79 #include <attr.h>
  80 #include <libcmdutils.h>
  81 #endif

  83 /* Trusted Extensions */
  84 #include <zone.h>
  85 #include <tsol/label.h>
  86 #include <sys/tsol/label_macro.h>

  88 #include "getresponse.h"
  89 /*
  90  * Source compatibility
  91  */

  93 /*
  94  * These constants come from archives.h and sys/fcntl.h
  95  * and were introduced by the extended attributes project
  96  * in Solaris 9.
  97  */
  98 #if !defined(O_XATTR)
  99 #define AT_SYMLINK_NOFOLLOW     0x1000
 100 #define AT_REMOVEDIR            0x1
 101 #define AT_FDCWD                0xffd19553
 102 #define _XATTR_HDRTYPE          'E'
 103 static int attropen();
 104 static int fstatat();
 105 static int renameat();
 106 static int unlinkat();
 107 static int openat();
 108 static int fchownat();
 109 static int futimesat();
 110 #endif

 112 /*
 113  * Compiling with -D_XPG4_2 gets this but produces other problems, so
 114  * instead of including sys/time.h and compiling with -D_XPG4_2, I'm
 115  * explicitly doing the declaration here.
 116  */
 117 int utimes(const char *path, const struct timeval timeval_ptr[]);

 119 #ifndef MINSIZE
 120 #define MINSIZE 250
 121 #endif
 122 #define DEF_FILE "/etc/default/tar"

 124 #define min(a, b)  ((a) < (b) ? (a) : (b))
 125 #define max(a, b)  ((a) > (b) ? (a) : (b))
```

```
 127 /* -DDEBUG      ONLY for debugging */
 128 #ifdef  DEBUG
 129 #undef  DEBUG
 130 #define DEBUG(a, b, c)\
 131         (void) fprintf(stderr, "DEBUG - "), (void) fprintf(stderr, a, b, c)
 132 #endif

 134 #define TBLOCK  512     /* tape block size--should be universal */

 136 #ifdef  BSIZE
 137 #define SYS_BLOCK BSIZE /* from sys/param.h:  secondary block size */
 138 #else   /* BSIZE */
 139 #define SYS_BLOCK 512   /* default if no BSIZE in param.h */
 140 #endif  /* BSIZE */

 142 #define NBLOCK  20
 143 #define NAMSIZ  100
 144 #define PRESIZ  155
 145 #define MAXNAM  256
 146 #define MODEMASK 0777777        /* file creation mode mask */
 147 #define POSIXMODES 07777        /* mask for POSIX mode bits */
 148 #define MAXEXT  9       /* reasonable max # extents for a file */
 149 #define EXTMIN  50      /* min blks left on floppy to split a file */

 151 /* max value dblock.dbuf.efsize can store */
 152 #define TAR_EFSIZE_MAX   0777777777

 154 /*
 155  * Symbols which specify the values at which the use of the 'E' function
 156  * modifier is required to properly store a file.
 157  *
 158  *     TAR_OFFSET_MAX    - the largest file size we can archive
 159  *     OCTAL7CHAR        - the limit for ustar gid, uid, dev
 160  */

 162 #ifdef XHDR_DEBUG
 163 /* tiny values which force the creation of extended header entries */
 164 #define TAR_OFFSET_MAX 9
 165 #define OCTAL7CHAR 2
 166 #else
 167 /* normal values */
 168 #define TAR_OFFSET_MAX  077777777777ULL
 169 #define OCTAL7CHAR      07777777
 170 #endif

 172 #define TBLOCKS(bytes)  (((bytes) + TBLOCK - 1) / TBLOCK)
 173 #define K(tblocks)      ((tblocks+1)/2) /* tblocks to Kbytes for printing */

 175 #define MAXLEV  (PATH_MAX / 2)
 176 #define LEV0    1
 177 #define SYMLINK_LEV0    0

 179 #define TRUE    1
 180 #define FALSE   0

 182 #define XATTR_FILE      1
 183 #define NORMAL_FILE     0

 185 #define PUT_AS_LINK     1
 186 #define PUT_NOTAS_LINK  0

 188 #ifndef VIEW_READONLY
 189 #define VIEW_READONLY   "SUNWattr_ro"
 190 #endif

 192 #ifndef VIEW_READWRITE
```



```
 193 #define VIEW_READWRITE  "SUNWattr_rw"
 194 #endif

 196 #if _FILE_OFFSET_BITS == 64
 197 #define FMT_off_t "lld"
 198 #define FMT_off_t_o "llo"
 199 #define FMT_blkcnt_t "lld"
 200 #else
 201 #define FMT_off_t "ld"
 202 #define FMT_off_t_o "lo"
 203 #define FMT_blkcnt_t "ld"
 204 #endif

 206 /* ACL support */

 208 static
 209 struct  sec_attr {
 210         char    attr_type;
 211         char    attr_len[7];
 212         char    attr_info[1];
 213 } *attr;
_____unchanged_portion_omitted_
 424 static  file_list_t     *exclude_tbl[TABLE_SIZE],
 425                         *include_tbl[TABLE_SIZE];

 427 static int      append_secattr(char **, int *, int, char *, char);
 428 static void     write_ancillary(union hblock *, char *, int, char);

 430 static void add_file_to_table(file_list_t *table[], char *str);
 431 static void assert_string(char *s, char *msg);
 432 static int istape(int fd, int type);
 433 static void backtape(void);
 434 static void build_table(file_list_t *table[], char *file);
 435 static int check_prefix(char **namep, char **dirp, char **compp);
 436 static void closevol(void);
 437 static void copy(void *dst, void *src);
 438 static int convtoreg(off_t);
 439 static void delete_target(int fd, char *comp, char *namep);
 440 static void doDirTimes(char *name, timestruc_t modTime);
 441 static void done(int n);
 442 static void dorep(char *argv[]);
 443 static void dotable(char *argv[]);
 444 static void doxtract(char *argv[]);
 445 static int tar_chdir(const char *path);
 446 static int is_directory(char *name);
 447 static int has_dot_dot(char *name);
 448 static int is_absolute(char *name);
 449 static char *make_relative_name(char *name, char **stripped_prefix);
 450 static void fatal(char *format, ...);
 451 static void vperror(int exit_status, char *fmt, ...);
 452 static void flushtape(void);
 453 static void getdir(void);
 454 static void *getmem(size_t);
 455 static void longt(struct stat *st, char aclchar);
 456 static void load_info_from_xtarhdr(u_longlong_t flag, struct xtar_hdr *xhdrp);
 457 static int makeDir(char *name);
 458 static void mterr(char *operation, int i, int exitcode);
 459 static void newvol(void);
 460 static void passtape(void);
 461 static void putempty(blkcnt_t n);
 462 static int putfile(char *longname, char *shortname, char *parent,
 463     attr_data_t *attrinfo, int filetype, int lev, int symlink_lev);
 464 static void readtape(char *buffer);
 465 static void seekdisk(blkcnt_t blocks);
 466 static void setPathTimes(int dirfd, char *path, timestruc_t modTime);
 467 static void setbytes_to_skip(struct stat *st, int err);
```

```
468 static void splitfile(char *longname, int ifd, char *name,
469         char *prefix, int filetype);
470 static void tomodes(struct stat *sp);
471 static void usage(void);
472 static int xblocks(int issysattr, off_t bytes, int ofile);
473 static int xsfile(int issysattr, int ofd);
474 static void resugname(int dirfd, char *name, int symflag);
475 static int bcheck(char *bstr);
476 static int checkdir(char *name);
477 static int checksum(union hblock *dblockp);
478 #ifdef  EUC
479 static int checksum_signed(union hblock *dblockp);
480 #endif  /* EUC */
481 static int checkupdate(char *arg);
482 static int checkw(char c, char *name);
483 static int cmp(char *b, char *s, int n);
484 static int defset(char *arch);
485 static int endtape(void);
486 static int is_in_table(file_list_t *table[], char *str);
487 static int notsame(void);
488 static int is_prefix(char *s1, char *s2);
489 static int response(void);
490 static int build_dblock(const char *, const char *, const char,
491         const int filetype, const struct stat *, const dev_t, const char *);
492 static unsigned int hash(char *str);

494 static blkcnt_t kcheck(char *kstr);
495 static off_t bsrch(char *s, int n, off_t l, off_t h);
496 static void onintr(int sig);
497 static void onquit(int sig);
498 static void onhup(int sig);
499 static uid_t getuidbyname(char *);
500 static gid_t getgidbyname(char *);
501 static char *getname(gid_t);
502 static char *getgroup(gid_t);
503 static int checkf(char *name, int mode, int howmuch);
504 static int writetbuf(char *buffer, int n);
505 static int wantit(char *argv[], char **namep, char **dirp, char **comp,
506     attr_data_t **attrinfo);
507 static void append_ext_attr(char *shortname, char **secinfo, int *len);
508 static int get_xdata(void);
509 static void gen_num(const char *keyword, const u_longlong_t number);
510 static void gen_date(const char *keyword, const timestruc_t time_value);
511 static void gen_string(const char *keyword, const char *value);
512 static void get_xtime(char *value, timestruc_t *xtime);
513 static int chk_path_build(char *name, char *longname, char *linkname,
514     char *prefix, char type, int filetype);
515 static int gen_utf8_names(const char *filename);
516 static int utf8_local(char *option, char **Xhdr_ptrptr, char *target,
517     const char *src, int max_val);
518 static int local_utf8(char **Xhdr_ptrptr, char *target, const char *src,
519     iconv_t iconv_cd, int xhdrflg, int max_val);
520 static int c_utf8(char *target, const char *source);
521 static int getstat(int dirfd, char *longname, char *shortname,
522     char *attrparent);
523 static void xattrs_put(char *, char *, char *, char *);
524 static void prepare_xattr(char **, char *, char *,
525     char, struct linkbuf *, int *);
526 static int put_link(char *name, char *longname, char *component,
527     char *longattrname, char *prefix, int filetype, char typeflag);
528 static int put_extra_attributes(char *longname, char *shortname,
529     char *longattrname, char *prefix, int filetype, char typeflag);
530 static int put_xattr_hdr(char *longname, char *shortname, char *longattrname,
531     char *prefix, int typeflag, int filetype, struct linkbuf *lp);
532 static int read_xattr_hdr(attr_data_t **attrinfo);
```



```
534 /* Trusted Extensions */
535 #define AUTO_ZONE       "/zone"

537 static void extract_attr(char **file_ptr, struct sec_attr *);
538 static int check_ext_attr(char *filename);
539 static void rebuild_comp_path(char *str, char **namep);
540 static int rebuild_lk_comp_path(char *str, char **namep);

542 static void get_parent(char *path, char *dir);
543 static char *get_component(char *path);
544 static int retry_open_attr(int pdirfd, int cwd, char *dirp, char *pattr,
545     char *name, int oflag, mode_t mode);
546 static char *skipslashes(char *string, char *start);
547 static void chop_endslashes(char *path);
548 static pid_t compress_file(void);
549 static void compress_back(void);
550 static void decompress_file(void);
551 static pid_t uncompress_file(void);
552 static void *compress_malloc(size_t);
553 static void check_compression(void);
554 static char *bz_suffix(void);
555 static char *gz_suffix(void);
556 static char *xz_suffix(void);
557 static char *add_suffix();
558 static void wait_pid(pid_t);
559 static void verify_compress_opt(const char *t);
560 static void detect_compress(void);

562 static  struct stat stbuf;

564 static  char    *myname;
565 static  char    *xtract_chdir = NULL;
566 static  int     checkflag = 0;
567 static  int     Xflag, Fflag, iflag, hflag, Bflag, Iflag;
568 static  int     rflag, xflag, vflag, tflag, mt, svmt, cflag, mflag, pflag;
569 static  int     uflag;
570 static  int     errflag;
571 static  int     oflag;
572 static  int     bflag, Aflag;
573 static  int     Pflag;                  /* POSIX conformant archive */
574 static  int     Eflag;                  /* Allow files greater than 8GB */
575 static  int     atflag;                 /* traverse extended attributes */
576 static  int     saflag;                 /* traverse extended sys attributes */
577 static  int     Dflag;                  /* Data change flag */
578 static  int     jflag;                  /* flag to use 'bzip2' */
579 static  int     zflag;                  /* flag to use 'gzip' */
580 static  int     Zflag;                  /* flag to use 'compress' */
581 static  int     Jflag;                  /* flag to use 'xz' */
582 static  int     aflag;                  /* flag to use autocompression */

584 /* Trusted Extensions */
585 static  int     Tflag;                  /* Trusted Extensions attr flags */
586 static  int     dir_flag;               /* for attribute extract */
587 static  int     mld_flag;               /* for attribute extract */
588 static  char    *orig_namep;            /* original namep - unadorned */
589 static  int     rpath_flag;             /* MLD real path is rebuilt */
590 static  char    real_path[MAXPATHLEN];  /* MLD real path */
591 static  int     lk_rpath_flag;          /* linked to real path is rebuilt */
592 static  char    lk_real_path[MAXPATHLEN]; /* linked real path */
593 static  bslabel_t       bs_label;       /* for attribute extract */
594 static  bslabel_t       admin_low;
595 static  bslabel_t       admin_high;
596 static  int     ignored_aprivs = 0;
597 static  int     ignored_fprivs = 0;
598 static  int     ignored_fattrs = 0;
```

```
600 static  int     term, chksum, wflag,
601                 first = TRUE, defaults_used = FALSE, linkerrok;
602 static  blkcnt_t        recno;
603 static  int     freemem = 1;
604 static  int     nblock = NBLOCK;
605 static  int     Errflg = 0;
606 static  int     exitflag = 0;

608 static  dev_t   mt_dev;         /* device containing output file */
609 static  ino_t   mt_ino;         /* inode number of output file */
610 static  int     mt_devtype;     /* dev type of archive, from stat structure */

612 static  int update = 1;         /* for 'open' call */

614 static  off_t   low;
615 static  off_t   high;

617 static  FILE    *tfile;
618 static  FILE    *vfile = stdout;
619 static  char    *tmpdir;
620 static  char    *tmp_suffix = "/tarXXXXXX";
621 static  char    *tname;
622 static  char    archive[] = "archive0=";
623 static  char    *Xfile;
624 static  char    *usefile;
625 static  char    tfname[1024];

627 static  int     mulvol;         /* multi-volume option selected */
628 static  blkcnt_t        blocklim; /* number of blocks to accept per volume */
629 static  blkcnt_t        tapepos; /* current block number to be written */
630 static  int     NotTape;        /* true if tape is a disk */
631 static  int     dumping;        /* true if writing a tape or other archive */
632 static  int     extno;          /* number of extent:  starts at 1 */
633 static  int     extotal;        /* total extents in this file */
634 static  off_t   extsize;        /* size of current extent during extraction */
635 static  ushort_t        Oumask = 0;     /* old umask value */
636 static  int is_posix;   /* true if archive we're reading is POSIX-conformant */
637 static  const   char    *magic_type = "ustar";
638 static  size_t  xrec_size = 8 * PATH_MAX;       /* extended rec initial size */
639 static  char    *xrec_ptr;
640 static  off_t   xrec_offset = 0;
641 static  int     Xhdrflag;
642 static  int     charset_type = 0;

644 static  u_longlong_t    xhdr_flgs;      /* Bits set determine which items */
645                                         /*   need to be in extended header. */
646 #define _X_DEVMAJOR     0x1
647 #define _X_DEVMINOR     0x2
648 #define _X_GID          0x4
649 #define _X_GNAME        0x8
650 #define _X_LINKPATH     0x10
651 #define _X_PATH         0x20
652 #define _X_SIZE         0x40
653 #define _X_UID          0x80
654 #define _X_UNAME        0x100
655 #define _X_ATIME        0x200
656 #define _X_CTIME        0x400
657 #define _X_MTIME        0x800
658 #define _X_XHDR         0x1000  /* Bit flag that determines whether 'X' */
659                                 /* typeflag was followed by 'A' or non 'A' */
660                                 /* typeflag. */
661 #define _X_LAST         0x40000000

663 #define PID_MAX_DIGITS          (10 * sizeof (pid_t) / 4)
664 #define TIME_MAX_DIGITS         (10 * sizeof (time_t) / 4)
665 #define LONG_MAX_DIGITS         (10 * sizeof (long) / 4)
```



```
666 #define ULONGLONG_MAX_DIGITS    (10 * sizeof (u_longlong_t) / 4)
667 /*
668  * UTF_8 encoding requires more space than the current codeset equivalent.
669  * Currently a factor of 2-3 would suffice, but it is possible for a factor
670  * of 6 to be needed in the future, so for saftey, we use that here.
671  */
672 #define UTF_8_FACTOR    6

674 static  u_longlong_t    xhdr_count = 0;
675 static char             xhdr_dirname[PRESIZ + 1];
676 static char             pidchars[PID_MAX_DIGITS + 1];
677 static char             *tchar = "";            /* null linkpath */

679 static  char    local_path[UTF_8_FACTOR * PATH_MAX + 1];
680 static  char    local_linkpath[UTF_8_FACTOR * PATH_MAX + 1];
681 static  char    local_gname[UTF_8_FACTOR * _POSIX_NAME_MAX + 1];
682 static  char    local_uname[UTF_8_FACTOR * _POSIX_NAME_MAX + 1];

684 /*
685  * The following mechanism is provided to allow us to debug tar in complicated
686  * situations, like when it is part of a pipe.  The idea is that you compile
687  * with -DWAITAROUND defined, and then add the 'D' function modifier to the
688  * target tar invocation, eg. "tar cDf tarfile file".  If stderr is available,
689  * it will tell you to which pid to attach the debugger; otherwise, use ps to
690  * find it.  Attach to the process from the debugger, and, *PRESTO*, you are
691  * there!
692  *
693  * Simply assign "waitaround = 0" once you attach to the process, and then
694  * proceed from there as usual.
695  */

697 #ifdef WAITAROUND
698 int waitaround = 0;             /* wait for rendezvous with the debugger */
699 #endif

701 #define BZIP            "/usr/bin/bzip2"
702 #define GZIP            "/usr/bin/gzip"
703 #define COMPRESS        "/usr/bin/compress"
704 #define XZ              "/usr/bin/xz"
705 #define BZCAT           "/usr/bin/bzcat"
706 #define GZCAT           "/usr/bin/gzcat"
707 #define ZCAT            "/usr/bin/zcat"
708 #define XZCAT           "/usr/bin/xzcat"
709 #define GSUF            8       /* number of valid 'gzip' sufixes */
710 #define BSUF            4       /* number of valid 'bzip2' sufixes */
711 #define XSUF            1       /* number of valid 'xz' suffixes */

713 static  char            *compress_opt;  /* compression type */

715 static  char            *gsuffix[] = {".gz", "-gz", ".z", "-z", "_z", ".Z",
716                         ".tgz", ".taz"};
717 static  char            *bsuffix[] = {".bz2", ".bz", ".tbz2", ".tbz"};
718 static  char            *xsuffix[] = {".xz"};
719 static  char            *suffix;


722 int
723 main(int argc, char *argv[])
724 {
725         char            *cp;
726         char            *tmpdirp;
727         pid_t           thispid;
728         pid_t           pid;
729         int             wstat;

731         (void) setlocale(LC_ALL, "");
```

```
732 #if !defined(TEXT_DOMAIN)       /* Should be defined by cc -D */
733 #define TEXT_DOMAIN "SYS_TEST"  /* Use this only if it weren't */
734 #endif
735         (void) textdomain(TEXT_DOMAIN);
736         if (argc < 2)
737                 usage();

739         tfile = NULL;
740         if ((myname = strdup(argv[0])) == NULL) {
741                 (void) fprintf(stderr, gettext(
742                     "tar: cannot allocate program name\n"));
743                 exit(1);
744         }

746         if (init_yes() < 0) {
747                 (void) fprintf(stderr, gettext(ERR_MSG_INIT_YES),
748                     strerror(errno));
749                 exit(2);
750         }

752         /*
753          *  For XPG4 compatibility, we must be able to accept the "--"
754          *  argument normally recognized by getopt; it is used to delimit
755          *  the end opt the options section, and so can only appear in
756          *  the position of the first argument.  We simply skip it.
757          */

759         if (strcmp(argv[1], "--") == 0) {
760                 argv++;
761                 argc--;
762                 if (argc < 3)
763                         usage();
764         }

766         argv[argc] = NULL;
767         argv++;

769         /*
770          * Set up default values.
771          * Search the operand string looking for the first digit or an 'f'.
772          * If you find a digit, use the 'archive#' entry in DEF_FILE.
773          * If 'f' is given, bypass looking in DEF_FILE altogether.
774          * If no digit or 'f' is given, still look in DEF_FILE but use '0'.
775          */
776         if ((usefile = getenv("TAPE")) == (char *)NULL) {
777                 for (cp = *argv; *cp; ++cp)
778                         if (isdigit(*cp) || *cp == 'f')
779                                 break;
780                 if (*cp != 'f') {
781                         archive[7] = (*cp)? *cp: '0';
782                         if (!(defaults_used = defset(archive))) {
783                                 usefile = NULL;
784                                 nblock = 1;
785                                 blocklim = 0;
786                                 NotTape = 0;
787                         }
788                 }
789         }

791         for (cp = *argv++; *cp; cp++)
792                 switch (*cp) {
793 #ifdef WAITAROUND
794                 case 'D':
795                         /* rendezvous with the debugger */
796                         waitaround = 1;
797                         break;
```



```
798 #endif
799                 case 'f':
800                         assert_string(*argv, gettext(
801                             "tar: tarfile must be specified with 'f' "
802                             "function modifier\n"));
803                         usefile = *argv++;
804                         break;
805                 case 'F':
806                         Fflag++;
807                         break;
808                 case 'c':
809                         cflag++;
810                         rflag++;
811                         update = 1;
812                         break;
813 #if defined(O_XATTR)
814                 case '@':
815                         atflag++;
816                         break;
817 #endif  /* O_XATTR */
818 #if defined(_PC_SATTR_ENABLED)
819                 case '/':
820                         saflag++;
821                         break;
822 #endif  /* _PC_SATTR_ENABLED */
823                 case 'u':
824                         uflag++;        /* moved code after signals caught */
825                         rflag++;
826                         update = 2;
827                         break;
828                 case 'r':
829                         rflag++;
830                         update = 2;
831                         break;
832                 case 'v':
833                         vflag++;
834                         break;
835                 case 'w':
836                         wflag++;
837                         break;
838                 case 'x':
839                         xflag++;
840                         break;
841                 case 'X':
842                         assert_string(*argv, gettext(
843                             "tar: exclude file must be specified with 'X' "
844                             "function modifier\n"));
845                         Xflag = 1;
846                         Xfile = *argv++;
847                         build_table(exclude_tbl, Xfile);
848                         break;
849                 case 't':
850                         tflag++;
851                         break;
852                 case 'm':
853                         mflag++;
854                         break;
855                 case 'p':
856                         pflag++;
857                         break;
858                 case 'D':
859                         Dflag++;
860                         break;
861                 case '-':
862                         /* ignore this silently */
863                         break;
```

```
864                 case '0':       /* numeric entries used only for defaults */
865                 case '1':
866                 case '2':
867                 case '3':
868                 case '4':
869                 case '5':
870                 case '6':
871                 case '7':
872                         break;
873                 case 'b':
874                         assert_string(*argv, gettext(
875                             "tar: blocking factor must be specified "
876                             "with 'b' function modifier\n"));
877                         bflag++;
878                         nblock = bcheck(*argv++);
879                         break;
880                 case 'n':               /* not a magtape (instead of 'k') */
881                         NotTape++;      /* assume non-magtape */
882                         break;
883                 case 'l':
884                         linkerrok++;
885                         break;
886                 case 'e':
887                         errflag++;
888                 case 'o':
889                         oflag++;
890                         break;
891                 case 'h':
892                         hflag++;
893                         break;
894                 case 'i':
895                         iflag++;
896                         break;
897                 case 'B':
898                         Bflag++;
899                         break;
900                 case 'P':
901                         Pflag++;
902                         break;
903                 case 'E':
904                         Eflag++;
905                         Pflag++;        /* Only POSIX archive made */
906                         break;
907                 case 'T':
908                         Tflag++;        /* Handle Trusted Extensions attrs */
909                         pflag++;        /* also set flag for ACL */
910                         break;
911                 case 'j':               /* compession "bzip2" */
912                         jflag = 1;
913                         break;
914                 case 'z':               /* compression "gzip" */
915                         zflag = 1;
916                         break;
917                 case 'Z':               /* compression "compress" */
918                         Zflag = 1;
919                         break;
920                 case 'J':               /* compression "xz" */
921                         Jflag = 1;
922                         break;
923                 case 'a':
924                         aflag = 1;      /* autocompression */
925                         break;
926                 default:
927                         (void) fprintf(stderr, gettext(
928                         "tar: %c: unknown function modifier\n"), *cp);
929                         usage();
```



```
930                 }

932         if (!rflag && !xflag && !tflag)
933                 usage();
934         if ((rflag && xflag) || (xflag && tflag) || (rflag && tflag)) {
935                 (void) fprintf(stderr, gettext(
936                 "tar: specify only one of [ctxru].\n"));
937                 usage();
938         }
939         if (cflag) {
940                 if ((jflag + zflag + Zflag + Jflag + aflag) > 1) {
941                         (void) fprintf(stderr, gettext(
942                             "tar: specify only one of [ajJzZ] to "
943                             "create a compressed file.\n"));
944                         usage();
945                 }
946         }
947         /* Trusted Extensions attribute handling */
948         if (Tflag && ((getzoneid() != GLOBAL_ZONEID) ||
949             !is_system_labeled())) {
950                 (void) fprintf(stderr, gettext(
951                 "tar: the 'T' option is only available with "
952                     "Trusted Extensions\nand must be run from "
953                     "the global zone.\n"));
954                 usage();
955         }
956         if (cflag && *argv == NULL)
957                 fatal(gettext("Missing filenames"));
958         if (usefile == NULL)
959                 fatal(gettext("device argument required"));

961         /* alloc a buffer of the right size */
962         if ((tbuf = (union hblock *)
963             calloc(sizeof (union hblock) * nblock, sizeof (char))) ==
964             (union hblock *)NULL) {
965                 (void) fprintf(stderr, gettext(
966                 "tar: cannot allocate physio buffer\n"));
967                 exit(1);
968         }

970         if ((xrec_ptr = malloc(xrec_size)) == NULL) {
971                 (void) fprintf(stderr, gettext(
972                     "tar: cannot allocate extended header buffer\n"));
973                 exit(1);
974         }

976 #ifdef WAITAROUND
977         if (waitaround) {
978                 (void) fprintf(stderr, gettext("Rendezvous with tar on pid"
979                     " %d\n"), getpid());

981                 while (waitaround) {
982                         (void) sleep(10);
983                 }
984         }
985 #endif

987         thispid = getpid();
988         (void) sprintf(pidchars, "%ld", thispid);
989         thispid = strlen(pidchars);

991         if ((tmpdirp = getenv("TMPDIR")) == (char *)NULL)
992                 (void) strcpy(xhdr_dirname, "/tmp");
993         else {
994                 /*
995                  * Make sure that dir is no longer than what can
```

```
 996                  * fit in the prefix part of the header.
 997                  */
 998                 if (strlen(tmpdirp) > (size_t)(PRESIZ - thispid - 12)) {
 999                         (void) strcpy(xhdr_dirname, "/tmp");
1000                         if ((vflag > 0) && (Eflag > 0))
1001                                 (void) fprintf(stderr, gettext(
1002                                     "Ignoring TMPDIR\n"));
1003                 } else
1004                         (void) strcpy(xhdr_dirname, tmpdirp);
1005         }
1006         (void) strcat(xhdr_dirname, "/PaxHeaders.");
1007         (void) strcat(xhdr_dirname, pidchars);

1009         if (rflag) {
1010                 if (cflag && usefile != NULL)  {
1011                         /* Set the compression type */
1012                         if (aflag)
1013                                 detect_compress();

1015                         if (jflag) {
1016                                 compress_opt = compress_malloc(strlen(BZIP)
1017                                     + 1);
1018                                 (void) strcpy(compress_opt, BZIP);
1019                         } else if (zflag) {
1020                                 compress_opt = compress_malloc(strlen(GZIP)
1021                                     + 1);
1022                                 (void) strcpy(compress_opt, GZIP);
1023                         } else if (Zflag) {
1024                                 compress_opt =
1025                                     compress_malloc(strlen(COMPRESS) + 1);
1026                                 (void) strcpy(compress_opt, COMPRESS);
1027                         } else if (Jflag) {
1028                                 compress_opt = compress_malloc(strlen(XZ) + 1);
1029                                 (void) strcpy(compress_opt, XZ);
1030                         }
1031                 } else {
1032                         /*
1033                          * Decompress if the file is compressed for
1034                          * an update or replace.
1035                          */
1036                         if (strcmp(usefile, "-") != 0) {
1037                                 check_compression();
1038                                 if (compress_opt != NULL) {
1039                                       decompress_file();
1040                                 }
1041                         }
1042                 }

1044                 if (cflag && tfile != NULL)
1045                         usage();
1046                 if (signal(SIGINT, SIG_IGN) != SIG_IGN)
1047                         (void) signal(SIGINT, onintr);
1048                 if (signal(SIGHUP, SIG_IGN) != SIG_IGN)
1049                         (void) signal(SIGHUP, onhup);
1050                 if (signal(SIGQUIT, SIG_IGN) != SIG_IGN)
1051                         (void) signal(SIGQUIT, onquit);
1052                 if (uflag) {
1053                         int tnum;
1054                         struct stat sbuf;

1056                         tmpdir = getenv("TMPDIR");
1057                         /*
1058                          * If the name is invalid or this isn't a directory,
1059                          * or the directory is not writable, then reset to
1060                          * a default temporary directory.
1061                          */
```



```
1062                         if (tmpdir == NULL || *tmpdir == '\0' ||
1063                             (strlen(tmpdir) + strlen(tmp_suffix)) > PATH_MAX) {
1064                                 tmpdir = "/tmp";
1065                         } else if (stat(tmpdir, &sbuf) < 0 ||
1066                             (sbuf.st_mode & S_IFMT) != S_IFDIR ||
1067                             (sbuf.st_mode & S_IWRITE) == 0) {
1068                                 tmpdir = "/tmp";
1069                         }

1071                         if ((tname = calloc(1, strlen(tmpdir) +
1072                             strlen(tmp_suffix) + 1)) == NULL) {
1073                                 vperror(1, gettext("tar: out of memory, "
1074                                     "cannot create temporary file\n"));
1075                         }
1076                         (void) strcpy(tname, tmpdir);
1077                         (void) strcat(tname, tmp_suffix);

1079                         if ((tnum = mkstemp(tname)) == -1)
1080                                 vperror(1, "%s", tname);
1081                         if ((tfile = fdopen(tnum, "w")) == NULL)
1082                                 vperror(1, "%s", tname);
1083                 }
1084                 if (strcmp(usefile, "-") == 0) {
1085                         if (cflag == 0)
1086                                 fatal(gettext(
1087                                 "can only create standard output archives."));
1088                         vfile = stderr;
1089                         mt = dup(1);
1090                         ++bflag;
1091                 } else {
1092                         if (cflag)
1093                                 mt = open(usefile,
1094                                     O_RDWR|O_CREAT|O_TRUNC, 0666);
1095                         else
1096                                 mt = open(usefile, O_RDWR);

1098                         if (mt < 0) {
1099                                 if (cflag == 0 || (mt =  creat(usefile, 0666))
1100                                     < 0)
1101                                 vperror(1, "%s", usefile);
1102                         }
1103                 }
1104                 /* Get inode and device number of output file */
1105                 (void) fstat(mt, &stbuf);
1106                 mt_ino = stbuf.st_ino;
1107                 mt_dev = stbuf.st_dev;
1108                 mt_devtype = stbuf.st_mode & S_IFMT;
1109                 NotTape = !istape(mt, mt_devtype);

1111                 if (rflag && !cflag && (mt_devtype == S_IFIFO))
1112                         fatal(gettext("cannot append to pipe or FIFO."));

1114                 if (Aflag && vflag)
1115                         (void) printf(
1116                         gettext("Suppressing absolute pathnames\n"));
1117                 if (cflag && compress_opt != NULL) {
1118                         pid = compress_file();
1119                         wait_pid(pid);
1120                 }
1121                 dorep(argv);
1122                 if (rflag && !cflag && (compress_opt != NULL))
1123                         compress_back();
1124         } else if (xflag || tflag) {
1125                 /*
1126                  * for each argument, check to see if there is a "-I file" pair.
1127                  * if so, move the 3rd argument into "-I"'s place, build_table()
```

```
1128                 * using "file"'s name and increment argc one (the second
1129                 * increment appears in the for loop) which removes the two
1130                 * args "-I" and "file" from the argument vector.
1131                 */
1132                for (argc = 0; argv[argc]; argc++) {
1133                        if (strcmp(argv[argc], "-I") == 0) {
1134                                if (!argv[argc+1]) {
1135                                        (void) fprintf(stderr, gettext(
1136                                        "tar: missing argument for -I flag\n"));
1137                                        done(2);
1138                                } else {
1139                                        Iflag = 1;
1140                                        argv[argc] = argv[argc+2];
1141                                        build_table(include_tbl, argv[++argc]);
1142                                }
1143                        } else if (strcmp(argv[argc], "-C") == 0) {
1144                                if (!argv[argc+1]) {
1145                                        (void) fprintf(stderr, gettext("tar: "
1146                                            "missing argument for -C flag\n"));
1147                                        done(2);
1148                                } else if (xtract_chdir != NULL) {
1149                                        (void) fprintf(stderr, gettext("tar: "
1150                                            "extract should have only one -C "
1151                                            "flag\n"));
1152                                        done(2);
1153                                } else {
1154                                        argv[argc] = argv[argc+2];
1155                                        xtract_chdir = argv[++argc];
1156                                }
1157                        }
1158                }
1159                if (strcmp(usefile, "-") == 0) {
1160                        mt = dup(0);
1161                        ++bflag;
1162                        /* try to recover from short reads when reading stdin */
1163                        ++Bflag;
1164                } else if ((mt = open(usefile, 0)) < 0)
1165                        vperror(1, "%s", usefile);

1167                /* Decompress if the file is compressed */

1169                if (strcmp(usefile, "-") != 0) {
1170                        check_compression();
1171                        if (compress_opt != NULL) {
1172                                pid = uncompress_file();
1173                                wait_pid(pid);
1174                        }
1175                }
1176                if (xflag) {
1177                        if (xtract_chdir != NULL) {
1178                                if (tar_chdir(xtract_chdir) < 0) {
1179                                        vperror(1, gettext("can't change "
1180                                            "directories to %s"), xtract_chdir);
1181                                }
1182                        }
1183                        if (Aflag && vflag)
1184                                (void) printf(gettext(
1185                                    "Suppressing absolute pathnames.\n"));

1187                        doxtract(argv);
1188                } else if (tflag)
1189                        dotable(argv);
1190        }
1191        else
1192                usage();
```



```
1194                done(Errflg);

1196                /* Not reached:  keep compiler quiet */
1197                return (1);
1198 }
```

_____***unchanged_portion_omitted_***

```
**********************************************************
   34828 Wed Jan 16 16:49:50 2013
new/usr/src/man/man1/tar.1
3474 tar should support -C on extract
Reviewed by: Robert Mustacchi <rm@joyent.com>
**********************************************************
'\" te
.\" Copyright 1989 AT&T
.\" Copyright (c) 2006, Sun Microsystems, Inc. All Rights Reserved
.\" Copyright 2012 Milan Jurik. All rights reserved.
.\" Copyright (c) 2013, Joyent, Inc. All rights reserved.
.\" Portions Copyright (c) 1992, X/Open Company Limited 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 TAR 1 "Jan 16, 2013"
.TH TAR 1 "May 9, 2012"
.SH NAME
tar \- create tape archives and add or extract files
.SH SYNOPSIS
.LP
.nf
\fBtar\fR c[BDeEFhilnopPqTvw@/[0-7]][bfk][X...][a|j|J|z|Z] [\fIblocksize\fR]
     [\fItarfile\fR] [\fIsize\fR] [\fIexclude-file\fR]...
     {\fIfile\fR | \(miI \fIinclude-file\fR | \(miC \fIdirectory\fR \fIfile\fR}.
.fi

.LP
.nf
\fBtar\fR r[BDeEFhilnqTvw@/[0-7]][bfk][j|J|z|Z] [\fIblocksize\fR] [\fItarfile\fR
     [\fIsize\fR]
     {\fIfile\fR | \(miI \fIinclude-file\fR | \(miC \fIdirectory\fR \fIfile\fR}.
.fi

.LP
.nf
\fBtar\fR t[BeFhilnqTv[0-7]][fk][X...][j|J|z|Z] [\fItarfile\fR] [\fIsize\fR]
     [\fIexclude-file\fR]... {\fIfile\fR | \(miI \fIinclude-file\fR}...
.fi

.LP
.nf
\fBtar\fR u[BDeEFhilnqTvw@/[0-7]][bfk][j|J|z|Z] [\fIblocksize\fR] [\fItarfile\fR
     [\fIsize\fR] \fIfile\fR...
.fi

.LP
.nf
\fBtar\fR x[BeFhilmnopqTvw@/[0-7]][fk][X...][j|J|z|Z] [\fItarfile\fR] [\fIsize\f
     [\fIexclude-file\fR]... [\(miC \fIdirectory\fR] [\fIfile\fR]...
     [\fIexclude-file\fR]... [\fIfile\fR]...
.fi

.SH DESCRIPTION
.sp
.LP
The \fBtar\fR command archives and extracts files to and from a single file
called a \fItarfile\fR. A tarfile is usually a magnetic tape, but it can be any
file. \fBtar\fR's actions are controlled by the \fIkey\fR argument. The
\fIkey\fR is a string of characters containing exactly one function letter
(\fBc\fR, \fBr\fR, \fBt\fR , \fBu\fR, or \fBx\fR) and zero or more function
modifiers (letters or digits), depending on the function letter used. The
```



```
\fIkey\fR string contains no SPACE characters. Function modifier arguments are
listed on the command line in the same order as their corresponding function
modifiers appear in the \fIkey\fR string.
.sp
.LP
The \fB\(miI\fR \fIinclude-file\fR, \fB\(miC\fR \fIdirectory file\fR, and
\fIfile\fR arguments specify which files or directories are to be archived or
extracted. In all cases, appearance of a directory name refers to the files and
(recursively) subdirectories of that directory. Arguments appearing within
braces (\fB{ }\fR) indicate that one of the arguments must be specified.
.SH OPERANDS
.sp
.LP
The following operands are supported:
.sp
.ne 2
.na
\fB\fB\(miC\fR \fIdirectory file\fR\fR
.ad
.sp .6
.RS 4n
Performs a \fBchdir\fR (see \fBcd\fR(1)) operation on \fIdirectory\fR and
performs the \fBc\fR (create) or \fBr\fR (replace) operation on \fIfile\fR. Use
short relative path names for \fIfile\fR. If \fIfile\fR is "\fB\&.\fR", archive
all files in \fIdirectory\fR. This operand enables archiving files from
multiple directories not related by a close common parent.
.sp
This option may also be passed once to \fBx\fR (extract).  In this case the
program will \fBchdir\fR to \fIdirectory\fR after opening the archive, but
before extracting its contents.
.RE

.sp
.ne 2
.na
\fB\fB\(miI\fR \fIinclude-file\fR\fR
.ad
.sp .6
.RS 4n
Opens \fIinclude-file\fR containing a list of files, one per line, and treats
it as if each file appeared separately on the command line. Be careful of
trailing white spaces. Also beware of leading white spaces, since, for each
line in the included file, the entire line (apart from the newline) is used to
match against the initial string of files to include. In the case where
excluded files (see \fBX\fR function modifier) are also specified, they take
precedence over all included files. If a file is specified in both the
\fIexclude-file\fR and the \fIinclude-file\fR (or on the command line), it is
excluded.
.RE

.sp
.ne 2
.na
\fB\fIfile\fR\fR
.ad
.sp .6
.RS 4n
A path name of a regular file or directory to be archived (when the \fBc\fR,
\fBr\fR or \fBu\fR functions are specified), extracted (\fBx\fR) or listed
(\fBt\fR). When \fIfile\fR is the path name of a directory, the action applies
to all of the files and (recursively) subdirectories of that directory.
.sp
When a file is archived, and the \fBE\fR flag (see \fBFunction Modifiers\fR) is
not specified, the filename cannot exceed 256 characters. In addition, it must
be possible to split the name between parent directory names so that the prefix
is no longer than 155 characters and the name is no longer than 100 characters.
```

```
125 If \fBE\fR is specified, a name of up to \fIPATH_MAX\fR characters can be
126 specified.
127 .sp
128 For example, a file whose basename is longer than 100 characters could not be
129 archived without using the \fBE\fR flag. A file whose directory portion is 200
130 characters and whose basename is 50 characters could be archived (without using
131 \fBE\fR) if a slash appears in the directory name somewhere in character
132 positions 151-156.
133 .RE

135 .SS "Function Letters"
136 .sp
137 .LP
138 The function portion of the key is specified by one of the following letters:
139 .sp
140 .ne 2
141 .na
142 \fB\fBc\fR\fR
143 .ad
144 .sp .6
145 .RS 4n
146 Create. Writing begins at the beginning of the tarfile, instead of at the end.
147 .RE

149 .sp
150 .ne 2
151 .na
152 \fB\fBr\fR\fR
153 .ad
154 .sp .6
155 .RS 4n
156 Replace. The named \fIfile\fRs are written at the end of the tarfile. A file
157 created with extended headers must be updated with extended headers (see
158 \fBE\fR flag under \fBFunction Modifiers\fR). A file created without extended
159 headers cannot be modified with extended headers.
160 .RE

162 .sp
163 .ne 2
164 .na
165 \fB\fBt\fR\fR
166 .ad
167 .sp .6
168 .RS 4n
169 Table of Contents. The names of the specified files are listed each time they
170 occur in the tarfile. If no \fIfile\fR argument is specified, the names of all
171 files and any associated extended attributes in the tarfile are listed. With
172 the \fBv\fR function modifier, additional information for the specified files
173 is displayed.
174 .RE

176 .sp
177 .ne 2
178 .na
179 \fB\fBu\fR\fR
180 .ad
181 .sp .6
182 .RS 4n
183 Update. The named \fIfile\fRs are written at the end of the tarfile if they are
184 not already in the tarfile, or if they have been modified since last written to
185 that tarfile. An update can be rather slow. A tarfile created on a 5.x system
186 cannot be updated on a 4.x system. A file created with extended headers must be
187 updated with extended headers (see \fBE\fR flag under \fBFunction
188 Modifiers\fR). A file created without extended headers cannot be modified with
189 extended headers.
190 .RE
```



```
192 .sp
193 .ne 2
194 .na
195 \fB\fBx\fR\fR
196 .ad
197 .sp .6
198 .RS 4n
199 Extract or restore. The named \fIfile\fRs are extracted from the tarfile and
200 written to the directory specified in the tarfile, relative to the current
201 directory. Use the relative path names of files and directories to be
202 extracted.
203 .sp
204 Absolute path names contained in the tar archive are unpacked using the
205 absolute path names, that is, the leading forward slash (\fB/\fR) is \fBnot\fR
206 stripped off.
207 .sp
208 If a named file matches a directory whose contents has been written to the
209 tarfile, this directory is recursively extracted. The owner, modification time,
210 and mode are restored (if possible); otherwise, to restore owner, you must be
211 the super-user. Character-special and block-special devices (created by
212 \fBmknod\fR(1M)) can only be extracted by the super-user. If no \fIfile\fR
213 argument is specified, the entire content of the tarfile is extracted. If the
214 tarfile contains several files with the same name, each file is written to the
215 appropriate directory, overwriting the previous one. Filename substitution
216 wildcards cannot be used for extracting files from the archive. Rather, use a
217 command of the form:
218 .sp
219 .in +2
220 .nf
221 \fBtar xvf ... /dev/rmt/0 \(gatar tf ... /dev/rmt/0 | \e
222      grep '\fIpattern\fR' \(ga\fR
223 .fi
224 .in -2
225 .sp

227 .RE

229 .sp
230 .LP
231 When extracting tapes created with the \fBr\fR or \fBu\fR functions, directory
232 modification times can not be set correctly. These same functions cannot be
233 used with many tape drives due to tape drive limitations such as the absence of
234 backspace or append capabilities.
235 .sp
236 .LP
237 When using the \fBr\fR, \fBu\fR, or \fBx\fR functions or the \fBX\fR function
238 modifier, the named files must match exactly the corresponding files in the
239 \fItarfile\fR. For example, to extract \fB\&./\fR\fIthisfile\fR, you must
240 specify \fB\&./\fR\fIthisfile,\fR and not \fIthisfile\fR. The \fBt\fR function
241 displays how each file was archived.
242 .SS "Function Modifiers"
243 .sp
244 .LP
245 The characters below can be used in conjunction with the letter that selects
246 the desired function.
247 .sp
248 .ne 2
249 .na
250 \fB\fBa\fR\fR
251 .ad
252 .sp .6
253 .RS 4n
254 During a \fBcreate\fR operation autodetect compression based on the archive
255 suffix.
256 .RE
```

```
.sp
.ne 2
.na
\fB\fBb\fR \fIblocksize\fR\fR
.ad
.sp .6
.RS 4n
Blocking Factor. Use when reading or writing to raw magnetic archives (see
\fBf\fR below). The \fIblocksize\fR argument specifies the number of 512-byte
tape blocks to be included in each read or write operation performed on the
tarfile. The minimum is \fB1\fR, the default is \fB20\fR. The maximum value is
a function of the amount of memory available and the blocking requirements of
the specific tape device involved (see \fBmtio\fR(7I) for details.) The maximum
cannot exceed \fBINT_MAX\fR/512 (\fB4194303\fR).
.sp
When a tape archive is being read, its actual blocking factor is automatically
detected, provided that it is less than or equal to the nominal blocking factor
(the value of the \fIblocksize\fR argument, or the default value if the \fBb\fR
modifier is not specified). If the actual blocking factor is greater than the
nominal blocking factor, a read error results. See Example 5 in EXAMPLES.
.RE

.sp
.ne 2
.na
\fB\fBB\fR\fR
.ad
.sp .6
.RS 4n
Block. Force \fBtar\fR to perform multiple reads (if necessary) to read exactly
enough bytes to fill a block. This function modifier enables \fBtar\fR to work
across the Ethernet, since pipes and sockets return partial blocks even when
more data is coming. When reading from standard input, "\fB\(mi\fR", this
function modifier is selected by default to ensure that \fBtar\fR can recover
from short reads.
.RE

.sp
.ne 2
.na
\fB\fBD\fR\fR
.ad
.sp .6
.RS 4n
Data change warnings. Used with \fBc\fR, \fBr\fR, or \fBu\fR function letters.
Ignored with \fBt\fR or \fBx\fR function letters. If the size of a file changes
while the file is being archived, treat this condition as a warning instead of
as an error. A warning message is still written, but the exit status is not
affected.
.RE

.sp
.ne 2
.na
\fB\fBe\fR\fR
.ad
.sp .6
.RS 4n
Error. Exit immediately with a positive exit status if any unexpected errors
occur. The \fBSYSV3\fR environment variable overrides the default behavior.
(See ENVIRONMENT VARIABLES section below.)
.RE

.sp
.ne 2
```



```
.na
\fB\fBE\fR\fR
.ad
.sp .6
.RS 4n
Write a tarfile with extended headers. (Used with \fBc\fR, \fBr\fR, or \fBu\fR
function letters. Ignored with \fBt\fR or \fBx\fR function letters.) When a
tarfile is written with extended headers, the modification time is maintained
with a granularity of microseconds rather than seconds. In addition, filenames
no longer than \fBPATH_MAX\fR characters that could not be archived without
\fBE\fR, and file sizes greater than \fB8GB\fR, are supported. The \fBE\fR flag
is required whenever the larger files and/or files with longer names, or whose
\fBUID/GID\fR exceed \fB2097151\fR, are to be archived, or if time granularity
of microseconds is desired.
.RE

.sp
.ne 2
.na
\fB\fBf\fR\fR
.ad
.sp .6
.RS 4n
File. Use the \fItarfile\fR argument as the name of the tarfile. If \fBf\fR is
specified, \fB/etc/default/tar\fR is not searched. If \fBf\fR is omitted,
\fBtar\fR uses the device indicated by the \fBTAPE\fR environment variable, if
set. Otherwise, \fBtar\fR uses the default values defined in
\fB/etc/default/tar\fR. The number matching the \fBarchive\fR\fIN\fR string is
used as the output device with the blocking and size specifications from the
file. For example,
.sp
.in +2
.nf
\fBtar -c 2/tmp/*\fR
.fi
.in -2
.sp

writes the output to the device specified as \fBarchive2\fR in
\fB/etc/default/tar\fR.
.sp
If the name of the tarfile is "\fB\(mi\fR", \fBtar\fR writes to the standard
output or reads from the standard input, whichever is appropriate. \fBtar\fR
can be used as the head or tail of a pipeline. \fBtar\fR can also be used to
move hierarchies with the command:
.sp
.in +2
.nf
example% \fBcd fromdir; tar cf \(mi .| (cd todir; tar xfBp \(mi)\fR
.fi
.in -2
.sp

.RE

.sp
.ne 2
.na
\fB\fBF\fR\fR
.ad
.sp .6
.RS 4n
With one \fBF\fR argument, \fBtar\fR excludes all directories named \fBSCCS\fR
and \fBRCS\fR from the tarfile. With two arguments, \fBFF\fR, \fBtar\fR
excludes all directories named SCCS and RCS, all files with \fB\&.o\fR as their
suffix, and all files named \fBerrs\fR, \fBcore\fR, and \fBa.out\fR. The
```

```
389 \fBSYSV3\fR environment variable overrides the default behavior. (See
390 ENVIRONMENT VARIABLES section below.)
391 .RE

393 .sp
394 .ne 2
395 .na
396 \fB\fBh\fR\fR
397 .ad
398 .sp .6
399 .RS 4n
400 Follow symbolic links as if they were normal files or directories. Normally,
401 \fBtar\fR does not follow symbolic links.
402 .RE

404 .sp
405 .ne 2
406 .na
407 \fB\fBi\fR\fR
408 .ad
409 .sp .6
410 .RS 4n
411 Ignore directory checksum errors.
412 .RE

414 .sp
415 .ne 2
416 .na
417 \fB\fBj\fR\fR
418 .ad
419 .sp .6
420 .RS 4n
421 Use \fBbzip2\fR for compressing or decompressing the archives.
422 .RE

424 .sp
425 .ne 2
426 .na
427 \fB\fBJ\fR\fR
428 .ad
429 .sp .6
430 .RS 4n
431 Use \fBxz\fR for compressing or decompressing the archives.
432 .RE

434 .sp
435 .ne 2
436 .na
437 \fB\fBk\fR \fIsize\fR\fR
438 .ad
439 .sp .6
440 .RS 4n
441 Requires \fBtar\fR to use the size argument as the size of an archive in
442 kilobytes. This is useful when the archive is intended for a fixed size device
443 such as floppy disks. Large files are then split across volumes if they do not
444 fit in the specified size.
445 .RE

447 .sp
448 .ne 2
449 .na
450 \fB\fBl\fR\fR
451 .ad
452 .sp .6
453 .RS 4n
454 Link. Output error message if unable to resolve all links to the files being
```



```
455 archived. If \fBl\fR is not specified, no error messages are printed.
456 .RE

458 .sp
459 .ne 2
460 .na
461 \fB\fBm\fR\fR
462 .ad
463 .sp .6
464 .RS 4n
465 Modify. The modification time of the file is the time of extraction. This
466 function modifier is valid only with the \fBx\fR function.
467 .RE

469 .sp
470 .ne 2
471 .na
472 \fB\fBn\fR\fR
473 .ad
474 .sp .6
475 .RS 4n
476 The file being read is a non-tape device. Reading of the archive is faster
477 since \fBtar\fR can randomly seek around the archive.
478 .RE

480 .sp
481 .ne 2
482 .na
483 \fB\fBo\fR\fR
484 .ad
485 .sp .6
486 .RS 4n
487 Ownership. Assign to extracted files the user and group identifiers of the user
488 running the program, rather than those on tarfile. This is the default behavior
489 for users other than root. If the \fBo\fR function modifier is not set and the
490 user is root, the extracted files takes on the group and user identifiers of
491 the files on tarfile (see \fBchown\fR(1) for more information). The \fBo\fR
492 function modifier is only valid with the \fBx\fR function.
493 .RE

495 .sp
496 .ne 2
497 .na
498 \fB\fBp\fR\fR
499 .ad
500 .sp .6
501 .RS 4n
502 Restore the named files to their original modes, and \fBACL\fRs if applicable,
503 ignoring the present \fBumask\fR(1). This is the default behavior if invoked as
504 super-user with the \fBx\fR function letter specified. If super-user,
505 \fBSETUID\fR, and sticky information are also extracted, and files are restored
506 with their original owners and permissions, rather than owned by root. When
507 this function modifier is used with the \fBc\fR function, \fBACL\fRs are
508 created in the tarfile along with other information. Errors occur when a
509 tarfile with \fBACL\fRs is extracted by previous versions of \fBtar\fR.
510 .RE

512 .sp
513 .ne 2
514 .na
515 \fB\fBP\fR\fR
516 .ad
517 .sp .6
518 .RS 4n
519 Suppress the addition of a trailing "\fB/\fR" on directory entries in the
520 archive.
```

```
.RE

.sp
.ne 2
.na
\fB\fBq\fR\fR
.ad
.sp .6
.RS 4n
Stop after extracting the first occurrence of the named file. \fBtar\fR
normally continues reading the archive after finding an occurrence of a file.
.RE

.sp
.ne 2
.na
\fB\fBT\fR\fR
.ad
.sp .6
.RS 4n
This modifier is only available if the system is configured with Trusted
Extensions.
.sp
When this modifier is used with the function letter \fBc\fR, \fBr,\fR or
\fBu\fR for creating, replacing or updating a tarfile, the sensitivity label
associated with each archived file and directory is stored in the tarfile.
.sp
Specifying \fBT\fR implies the function modifier \fBp\fR.
.sp
When used with the function letter \fBx\fR for extracting a tarfile, the tar
program verifies that the file's sensitivity label specified in the archive
equals the sensitivity label of the destination directory. If not, the file is
not restored. This operation must be invoked from the global zone. If the
archived file has a relative pathname, it is restored to the corresponding
directory with the same label, if available. This is done by prepending to the
current destination directory the root pathname of the zone whose label equals
the file. If no such zone exists, the file is not restored.
.sp
Limited support is provided for extracting labeled archives from Trusted
Solaris 8. Only sensitivity labels, and multi-level directory specifications
are interpreted. Privilege specifications and audit attribute flags are
silently ignored. Multilevel directory specifications including symbolic links
to single level directories are are mapped into zone-relative pathnames if a
zone with the same label is available. This support is intended to facilitate
migration of home directories. Architectural differences preclude the
extraction of arbitrarily labeled files from Trusted Solaris 8 into identical
pathnames in Trusted Extensions. Files cannot be extracted unless their
archived label matches the destination label.
.RE

.sp
.ne 2
.na
\fB\fBv\fR\fR
.ad
.sp .6
.RS 4n
Verbose. Output the name of each file preceded by the function letter. With the
\fBt\fR function, \fBv\fR provides additional information about the tarfile
entries. The listing is similar to the format produced by the \fB-l\fR option
of the \fBls\fR(1) command.
.RE

.sp
.ne 2
.na
```



```
\fB\fBw\fR\fR
.ad
.sp .6
.RS 4n
What. Output the action to be taken and the name of the file, then await the
user's confirmation. If the response is affirmative, the action is performed;
otherwise, the action is not performed. This function modifier cannot be used
with the \fBt\fR function.
.RE

.sp
.ne 2
.na
\fB\fBX\fR\fR
.ad
.sp .6
.RS 4n
Exclude. Use the \fIexclude-file\fR argument as a file containing a list of
relative path names for files (or directories) to be excluded from the tarfile
when using the functions \fBc\fR, \fBx\fR, or \fBt\fR. Be careful of trailing
white spaces. Also beware of leading white spaces, since, for each line in the
excluded file, the entire line (apart from the newline) is used to match
against the initial string of files to exclude. Lines in the exclude file are
matched exactly, so an entry like "\fB/var\fR" does \fBnot\fR exclude the
\fB/var\fR directory if \fBtar\fR is backing up relative pathnames. The entry
should read "\fB\&./var\fR" under these circumstances. The \fBtar\fR command
does not expand shell metacharacters in the exclude file, so specifying entries
like "\fB*.o\fR" does not have the effect of excluding all files with names
suffixed with "\fB\&.o\fR". If a complex list of files is to be excluded, the
exclude file should be generated by some means such as the \fBfind\fR(1)
command with appropriate conditions.
.sp
Multiple \fBX\fR arguments can be used, with one \fIexclude-file\fR per
argument. In the case where included files (see \fB\(miI\fR \fIinclude-file\fR
operand) are also specified, the excluded files take precedence over all
included files. If a file is specified in both the \fIexclude-file\fR and the
\fIinclude-file\fR (or on the command line), it is excluded.
.RE

.sp
.ne 2
.na
\fB\fBz\fR\fR
.ad
.sp .6
.RS 4n
Use \fBgzip\fR for compressing or decompressing the archives.
.RE

.sp
.ne 2
.na
\fB\fBZ\fR\fR
.ad
.sp .6
.RS 4n
Use \fBcompress\fR for compressing or decompressing the archives.
.RE

.sp
.ne 2
.na
\fB\fB@\fR\fR
.ad
.sp .6
.RS 4n
```

```
Include extended attributes in archive. By default, \fBtar\fR does not place
extended attributes in the archive. With this flag, \fBtar\fR looks for
extended attributes on the files to be placed in the archive and add them to
the archive. Extended attributes go in the archive as special files with a
special type label. When this modifier is used with the \fBx\fR function,
extended attributes are extracted from the tape along with the normal file
data. Extended attribute files can only be extracted from an archive as part of
a normal file extract. Attempts to explicitly extract attribute records are
ignored.
.RE

.sp
.ne 2
.na
\fB\fB/\fR\fR
.ad
.sp .6
.RS 4n
Include extended system attributes in archive. By default, \fBtar\fR does not
place extended system attributes in the archive. With this flag, \fBtar\fR
looks for extended system attributes on the files to be placed in the archive
and adds them to the archive. Extended system attributes go in the archive as
special files with a special type label. When this modifier is used with the
\fBx\fR function, extended system attributes are extracted from the tape along
with the normal file data. Extended system attribute files can only be
extracted from an archive as part of a normal file extract. Attempts to
explicitly extract attribute records are ignored.
.RE

.sp
.ne 2
.na
\fB\fB[0-7]\fR\fR
.ad
.sp .6
.RS 4n
Select an alternative drive on which the tape is mounted. The default entries
are specified in \fB/etc/default/tar\fR. If no digit or \fBf\fR function
modifier is specified, the entry in \fB/etc/default/tar\fR with digit "\fB0\fR"
is the default.
.RE

.SH USAGE
.sp
.LP
See \fBlargefile\fR(5) for the description of the behavior of \fBtar\fR when
encountering files greater than or equal to 2 Gbyte ( 2^31 bytes).
.sp
.LP
The automatic determination of the actual blocking factor can be fooled when
reading from a pipe or a socket (see the \fBB\fR function modifier below).
.sp
.LP
1/4" streaming tape has an inherent blocking factor of one 512-byte block. It
can be read or written using any blocking factor.
.sp
.LP
This function modifier works for archives on disk files and block special
devices, among others, but is intended principally for tape devices.
.sp
.LP
For information on \fBtar\fR header format, see \fBarchives.h\fR(3HEAD).
.SH EXAMPLES
.LP
\fBExample 1 \fRCreating an archive of your home directory
.sp
```



```
.LP
The following is an example using \fBtar\fR to create an archive of your home
directory on a tape mounted on drive \fB/dev/rmt/0\fR:

.sp
.in +2
.nf
example% \fBcd\fR
example% \fBtar cvf /dev/rmt/0\fR .
\fImessages from\fR tar
.fi
.in -2
.sp

.sp
.LP
The \fBc\fR function letter means create the archive. The \fBv\fR function
modifier outputs messages explaining what \fBtar\fR is doing. The \fBf\fR
function modifier indicates that the tarfile is being specified
(\fB/dev/rmt/0\fR in this example). The dot (\fB\&.\fR) at the end of the
command line indicates the current directory and is the argument of the \fBf\fR
function modifier.

.sp
.LP
Display the table of contents of the tarfile with the following command:

.sp
.in +2
.nf
example% \fBtar tvf /dev/rmt/0\fR
.fi
.in -2
.sp

.sp
.LP
The output is similar to the following for the POSIX locale:

.sp
.in +2
.nf
rw\(mir\(mi\(mir\(mi\(mi   1677/40    2123    Nov  7 18:15 1985    ./test.c
\&...
example%
.fi
.in -2
.sp

.sp
.LP
The columns have the following meanings:

.RS +4
.TP
.ie t \(bu
.el o
column 1 is the access permissions to \fB\&./test.c\fR
.RE
.RS +4
.TP
.ie t \(bu
.el o
column 2 is the \fIuser-id\fR/\fIgroup-id\fR of \fB\&./test.c\fR
.RE
.RS +4
```

```
.TP
.ie t \(bu
.el o
column 3 is the size of \fB\&./test.c\fR in bytes
.RE
.RS +4
.TP
.ie t \(bu
.el o
column 4 is the modification date of \fB\&./test.c\fR. When the \fBLC_TIME\fR
category is not set to the POSIX locale, a different format and date order
field can be used.
.RE
.RS +4
.TP
.ie t \(bu
.el o
column 5 is the name of \fB\&./test.c\fR
.RE
.sp
.LP
To extract files from the archive:

.sp
.in +2
.nf
example% \fBtar xvf /dev/rmt/0\fR
\fImessages from\fR tar
example%
.fi
.in -2
.sp

.sp
.LP
If there are multiple archive files on a tape, each is separated from the
following one by an EOF marker. To have \fBtar\fR read the first and second
archives from a tape with multiple archives on it, the \fInon-rewinding\fR
version of the tape device name must be used with the \fBf\fR function
modifier, as follows:

.sp
.in +2
.nf
example% \fBtar xvfp /dev/rmt/0n \fIread first archive from tape\fR\fR
\fImessages from\fR tar
example% \fBtar xvfp /dev/rmt/0n \fIread second archive from tape\fR\fR
\fImessages from\fR tar
example%
.fi
.in -2
.sp

.sp
.LP
Notice that in some earlier releases, the above scenario did not work
correctly, and intervention with \fBmt\fR(1) between \fBtar\fR invocations was
necessary. To emulate the old behavior, use the non-rewind device name
containing the letter \fBb\fR for BSD behavior. See the \fBClose Operations\fR
section of the \fBmtio\fR(7I) manual page.

.LP
\fBExample 2 \fRArchiving files from /usr/include and from /etc to default tape
drive 0
.sp
.LP
```



```
To archive files from \fB/usr/include\fR and from \fB/etc\fR to default tape
drive \fB0\fR:

.sp
.in +2
.nf
example% \fBtar c -C /usr include -C /etc .\fR
.fi
.in -2
.sp

.sp
.LP
The table of contents from the resulting tarfile would produce output like the
following:

.sp
.in +2
.nf
include/
include/a.out.h
\fIand all the other files in\fR \fB/usr/include ...\fR
\&./chown \fIand all the other files in\fR /etc
.fi
.in -2
.sp

.sp
.LP
To extract all files in the \fBinclude\fR directory:

.sp
.in +2
.nf
example% \fBtar xv include
x include/, 0 bytes, 0 tape blocks \e
    \fIand all files under\fR include ...\fR
.fi
.in -2
.sp

.LP
\fBExample 3 \fRTransferring files across the network
.sp
.LP
The following is an example using \fBtar\fR to transfer files across the
network. First, here is how to archive files from the local machine
(\fBexample\fR) to a tape on a remote system (\fBhost\fR):

.sp
.in +2
.nf
example% \fBtar cvfb \(mi 20 \fIfiles\fR| \e
    rsh \fIhost\fR dd of=/dev/rmt/0 obs=20b\fR
\fImessages from\fR tar
example%
.fi
.in -2
.sp

.sp
.LP
In the example above, we are \fIcreating\fR a \fItarfile\fR with the \fBc\fR
key letter, asking for \fIverbose\fR output from \fBtar\fR with the \fBv\fR
function modifier, specifying the name of the output \fItarfile\fR using the
\fBf\fR function modifier (the standard output is where the \fItarfile\fR
```

```
 917 appears, as indicated by the ‘\fB\(mi\fR\&’ sign), and specifying the blocksize
 918 (\fB20\fR) with the \fBb\fR function modifier. If you want to change the
 919 blocksize, you must change the blocksize arguments both on the \fBtar\fR
 920 command \fIand\fR on the \fBdd\fR command.

 922 .LP
 923 \fBExample 4 \fRRetrieving files from a tape on the remote system back to the
 924 local system
 925 .sp
 926 .LP
 927 The following is an example that uses \fBtar\fR to retrieve files from a tape
 928 on the remote system back to the local system:

 930 .sp
 931 .in +2
 932 .nf
 933 example% \fBrsh -n host dd if=/dev/rmt/0 bs=20b | \e
 934     tar xvBfb \(mi 20 \fIfiles\fR\fR
 935 \fImessages from\fR tar
 936 example%
 937 .fi
 938 .in -2
 939 .sp

 941 .sp
 942 .LP
 943 In the example above, we are \fIextracting\fR from the \fItarfile\fR with the
 944 \fBx\fR key letter, asking for \fIverbose\fR \fIoutput\fR \fIfrom\fR \fBtar\fR
 945 with the \fBv\fR function modifier, telling \fBtar\fR it is reading from a pipe
 946 with the \fBB\fR function modifier, specifying the name of the input
 947 \fItarfile\fR using the \fBf\fR function modifier (the standard input is where
 948 the \fItarfile\fR appears, as indicated by the "\fB\(mi\fR" sign), and
 949 specifying the blocksize (\fB20\fR) with the \fBb\fR function modifier.

 951 .LP
 952 \fBExample 5 \fRCreating an archive of the home directory
 953 .sp
 954 .LP
 955 The following example creates an archive of the home directory on
 956 \fB/dev/rmt/0\fR with an actual blocking factor of \fB19\fR:

 958 .sp
 959 .in +2
 960 .nf
 961 example% \fBtar cvfb /dev/rmt/0 19 $HOME\fR
 962 .fi
 963 .in -2
 964 .sp

 966 .sp
 967 .LP
 968 To recognize this archive’s actual blocking factor without using the \fBb\fR
 969 function modifier:

 971 .sp
 972 .in +2
 973 .nf
 974 example% \fBtar tvf /dev/rmt/0\fR
 975 tar: blocksize = 19
 976 \&...
 977 .fi
 978 .in -2
 979 .sp

 981 .sp
 982 .LP
```



```
 983 To recognize this archive’s actual blocking factor using a larger nominal
 984 blocking factor:

 986 .sp
 987 .in +2
 988 .nf
 989 example% \fBtar tvf /dev/rmt/0 30\fR
 990 tar: blocksize = 19
 991 \&...
 992 .fi
 993 .in -2
 994 .sp

 996 .sp
 997 .LP
 998 Attempt to recognize this archive’s actual blocking factor using a nominal
 999 blocking factor that is too small:

1001 .sp
1002 .in +2
1003 .nf
1004 example% \fBtar tvf /dev/rmt/0 10\fR
1005 tar: tape read error
1006 .fi
1007 .in -2
1008 .sp

1010 .SH ENVIRONMENT VARIABLES
1011 .sp
1012 .LP
1013 See \fBenviron\fR(5) for descriptions of the following environment variables
1014 that affect the execution of \fBtar\fR: \fBLC_COLLATE\fR, \fBLC_CTYPE\fR,
1015 \fBLC_MESSAGES\fR, \fBLC_TIME\fR, \fBTZ\fR, and \fBNLSPATH\fR.
1016 .sp
1017 .LP
1018 Affirmative responses are processed using the extended regular expression
1019 defined for the \fByesexpr\fR keyword in the \fBLC_MESSAGES\fR category of the
1020 user’s locale. The locale specified in the \fBLC_COLLATE\fR category defines
1021 the behavior of ranges, equivalence classes, and multi-character collating
1022 elements used in the expression defined for \fByesexpr\fR. The locale specified
1023 in \fBLC_CTYPE\fR determines the locale for interpretation of sequences of
1024 bytes of text data a characters, the behavior of character classes used in the
1025 expression defined for the \fByesexpr\fR. See \fBlocale\fR(5).
1026 .SH EXIT STATUS
1027 .sp
1028 .LP
1029 The following exit values are returned:
1030 .sp
1031 .ne 2
1032 .na
1033 \fB\fB0\fR\fR
1034 .ad
1035 .sp .6
1036 .RS 4n
1037 Successful completion.
1038 .RE

1040 .sp
1041 .ne 2
1042 .na
1043 \fB\fB>0\fR\fR
1044 .ad
1045 .sp .6
1046 .RS 4n
1047 An error occurred.
1048 .RE
```

```
1050 .SH FILES
1051 .sp
1052 .ne 2
1053 .na
1054 \fB\fB/dev/rmt/[0-7][b][n]\fR\fR
1055 .ad
1056 .sp .6
1057 .RS 4n

1059 .RE

1061 .sp
1062 .ne 2
1063 .na
1064 \fB\fB/dev/rmt/[0-7]l[b][n]\fR\fR
1065 .ad
1066 .sp .6
1067 .RS 4n

1069 .RE

1071 .sp
1072 .ne 2
1073 .na
1074 \fB\fB/dev/rmt/[0-7]m[b][n]\fR\fR
1075 .ad
1076 .sp .6
1077 .RS 4n

1079 .RE

1081 .sp
1082 .ne 2
1083 .na
1084 \fB\fB/dev/rmt/[0-7]h[b][n]\fR\fR
1085 .ad
1086 .sp .6
1087 .RS 4n

1089 .RE

1091 .sp
1092 .ne 2
1093 .na
1094 \fB\fB/dev/rmt/[0-7]u[b][n]\fR\fR
1095 .ad
1096 .sp .6
1097 .RS 4n

1099 .RE

1101 .sp
1102 .ne 2
1103 .na
1104 \fB\fB/dev/rmt/[0-7]c[b][n]\fR\fR
1105 .ad
1106 .sp .6
1107 .RS 4n

1109 .RE

1111 .sp
1112 .ne 2
1113 .na
1114 \fB\fB/etc/default/tar\fR\fR
```



```
1115 .ad
1116 .sp .6
1117 .RS 4n
1118 Settings might look like this:
1119 .br
1120 .in +2
1121 \fBarchive0=/dev/rmt/0\fR
1122 .in -2
1123 .br
1124 .in +2
1125 \fBarchive1=/dev/rmt/0n\fR
1126 .in -2
1127 .br
1128 .in +2
1129 \fBarchive2=/dev/rmt/1\fR
1130 .in -2
1131 .br
1132 .in +2
1133 \fBarchive3=/dev/rmt/1n\fR
1134 .in -2
1135 .br
1136 .in +2
1137 \fBarchive4=/dev/rmt/0\fR
1138 .in -2
1139 .br
1140 .in +2
1141 \fBarchive5=/dev/rmt/0n\fR
1142 .in -2
1143 .br
1144 .in +2
1145 \fBarchive6=/dev/rmt/1\fR
1146 .in -2
1147 .br
1148 .in +2
1149 \fBarchive7=/dev/rmt/1n\fR
1150 .in -2
1151 .RE

1153 .sp
1154 .ne 2
1155 .na
1156 \fB\fB/tmp/tar*\fR\fR
1157 .ad
1158 .sp .6
1159 .RS 4n

1161 .RE

1163 .SH ATTRIBUTES
1164 .sp
1165 .LP
1166 See \fBattributes\fR(5) for descriptions of the following attributes:
1167 .sp

1169 .sp
1170 .TS
1171 box;
1172 c | c
1173 l | l .
1174 ATTRIBUTE TYPE  ATTRIBUTE VALUE
1175 _
1176 CSI     Enabled
1177 _
1178 Interface Stability     Committed
1179 .TE
```

```
.SH SEE ALSO
.sp
.LP
\fBar\fR(1), \fBbasename\fR(1), \fBbzip2\fR(1), \fBcd\fR(1), \fBchown\fR(1),
\fBcompress\fR)(1), \fBcpio\fR(1), \fBcsh\fR(1), \fBdirname\fR(1),
\fBfind\fR(1), \fBgzip\fR(1), \fBls\fR(1), \fBmt\fR(1), \fBpax\fR(1),
\fBsetfacl\fR(1), \fBumask\fR(1), \fBxz\fR(1), \fBmknod\fR(1M),
\fBarchives.h\fR(3HEAD), \fBattributes\fR(5), \fBenviron\fR(5),
\fBfsattr\fR(5), \fBlargefile\fR(5), \fBmtio\fR(7I)
.SH DIAGNOSTICS
.sp
.LP
Diagnostic messages are output for bad key characters and tape read/write
errors, and for insufficient memory to hold the link tables.
.SH NOTES
.sp
.LP
There is no way to access the \fIn\fR-th occurrence of a file.
.sp
.LP
Tape errors are handled ungracefully.
.sp
.LP
The \fBtar\fR archive format allows \fBUID\fRs and \fBGID\fRs up to
\fB2097151\fR to be stored in the archive header. Files with \fBUID\fRs and
\fBGID\fRs greater than this value is archived with the \fBUID\fR and \fBGID\fR
of \fB60001\fR.
.sp
.LP
If an archive is created that contains files whose names were created by
processes running in multiple locales, a single locale that uses a full 8-bit
codeset (for example, the \fBen_US\fR locale) should be used both to create the
archive and to extract files from the archive.
.sp
.LP
Neither the \fBr\fR function letter nor the \fBu\fR function letter can be used
with quarter-inch archive tapes, since these tape drives cannot backspace.
.sp
.LP
Since \fBtar\fR has no options, the standard "\fB\(mi\(mi\fR" argument that is
normally used in other utilities to terminate recognition of options is not
needed. If used, it is recognized only as the first argument and is ignored.
.sp
.LP
Since \fB\(miC\fR \fIdirectory\fR \fIfile\fR and \fB\(miI\fR \fIinclude-file\fR
are multi-argument operands, any of the following methods can be used to
archive or extract a file named \fB\(miC\fR or \fB\(miI\fR:
.RS +4
.TP
1.
Specify them using file operands containing a \fB/\fR character on the
command line (such as \fB/home/joe/\(miC\fR or \fB\&./\(miI\fR).
.RE
.RS +4
.TP
2.
Include them in an include file with \fB\(miI\fR \fIinclude-file\fR.
.RE
.RS +4
.TP
3.
Specify the directory in which the file resides:
.sp
.in +2
.nf
\fB-C \fIdirectory\fR -C\fR
```



```
.fi
.in -2
.sp

or
.sp
.in +2
.nf
\fB-C \fIdirectory\fR -I\fR
.fi
.in -2
.sp

.RE
.RS +4
.TP
4.
Specify the entire directory in which the file resides:
.sp
.in +2
.nf
\fB-C \fIdirectory\fR .\fR
.fi
.in -2
.sp

.RE
```