TAR Format
Intel byte order

Information from File Format List 2.0 by Max Maischein.

--------!-CONTACT_INFO----------------------
If you notice any mistakes or omissions, please let me know! It is only with YOUR help that the list can continue to grow. Please send all changes to me rather than distributing a modified version of the list.

This file has been authored in the style of the INTERxxy.* file list by Ralf Brown, and uses almost the same format.

Please read the file FILEFMTS.1ST before asking me any questions. You may find
that they have already been addressed.

Max Maischein

Max Maischein, 2:244/1106.17
Max_Maischein@spam.fido.de
corion@informatik.uni-frankfurt.de
Corion on #coders@IRC
--------!-DISCLAIMER------------------------
DISCLAIMER: THIS MATERIAL IS PROVIDED "AS IS". I verify the information contained in this list to the best of my ability, but I cannot be held responsible for any problems caused by use or misuse of the information, especially for those file formats foreign to the PC, like AMIGA or SUN file formats. If an information it is marked "guesswork" or undocumented, you should check it carefully to make sure your program will not break with an unexpected value (and please let me know whether or not it works the same way).

Information marked with "???" is known to be incomplete or guesswork.

Some file formats were not released by their creators, others are regarded as proprietary, which means that if your programs deal with them, you might be looking for trouble. I don't care about this.

The Unix TAR program is an archiver program which stores files in a single archive without compression.
OFFSET Count TYPE Description
@section The Standard Format
A @dfn{tar tape} or file contains a series of records. Each record contains @code{RECORDSIZE} bytes. Although this format may be thought of as being on magnetic tape, other media are often used.

Each file archived is represented by a header record which describes the file, followed by zero or more records which give the contents of the file. At the end of the archive file there may be a record filled with binary zeros as an end-of-file marker. A reasonable

system should write a record of zeros at the end, but must not assume that such a record exists when reading an archive.

The records may be @dfn{blocked} for physical I/O operations. Each block of @var{N} records (where @var{N} is set by the @samp{-b} option to @code{tar}) is written with a single @code{write()} operation. On magnetic tapes, the result of such a write is a single tape record. When writing an archive, the last block of records should be written at the full size, with records after the zero record containing all zeroes. When reading an archive, a reasonable system should properly handle an archive whose last block is shorter than the rest, or which contains garbage records after a zero record.

The header record is defined in C as follows:

@example
```
/*
 * Standard Archive Format - Standard TAR - USTAR
 */
#define  RECORDSIZE  512
#define  NAMSIZ      100
#define  TUNMLEN      32
#define  TGNMLEN      32

union record @{
    char        charptr[RECORDSIZE];
    struct header @{
        char    name[NAMSIZ];
        char    mode[8];
        char    uid[8];
        char    gid[8];
        char    size[12];
        char    mtime[12];
        char    chksum[8];
        char    linkflag;
        char    linkname[NAMSIZ];
        char    magic[8];
        char    uname[TUNMLEN];
        char    gname[TGNMLEN];
        char    devmajor[8];
        char    devminor[8];
    @} header;
@};

/* The checksum field is filled with this while the checksum is computed.
*/
#define    CHKBLANKS    "        "        /* 8 blanks, no null */

/* The magic field is filled with this if uname and gname are valid. */
#define    TMAGIC    "ustar  "        /* 7 chars and a null */
```

```
/* The magic field is filled with this if this is a GNU format dump entry
*/
#define    GNUMAGIC  "GNUtar "        /* 7 chars and a null */

/* The linkflag defines the type of file */
#define  LF_OLDNORMAL '\0'       /* Normal disk file, Unix compatible */
#define  LF_NORMAL    '0'        /* Normal disk file */
#define  LF_LINK      '1'        /* Link to previously dumped file */
#define  LF_SYMLINK   '2'        /* Symbolic link */
#define  LF_CHR       '3'        /* Character special file */
#define  LF_BLK       '4'        /* Block special file */
#define  LF_DIR       '5'        /* Directory */
#define  LF_FIFO      '6'        /* FIFO special file */
#define  LF_CONTIG    '7'        /* Contiguous file */

/* Further link types may be defined later. */

/* Bits used in the mode field - values in octal */
#define  TSUID    04000        /* Set UID on execution */
#define  TSGID    02000        /* Set GID on execution */
#define  TSVTX    01000        /* Save text (sticky bit) */

/* File permissions */
#define  TUREAD   00400        /* read by owner */
#define  TUWRITE  00200        /* write by owner */
#define  TUEXEC   00100        /* execute/search by owner */
#define  TGREAD   00040        /* read by group */
#define  TGWRITE  00020        /* write by group */
#define  TGEXEC   00010        /* execute/search by group */
#define  TOREAD   00004        /* read by other */
#define  TOWRITE  00002        /* write by other */
#define  TOEXEC   00001        /* execute/search by other */
@end example

All characters in header records are represented by using 8-bit
characters in the local variant of ASCII.  Each field within the
structure is contiguous; that is, there is no padding used within
the structure.  Each character on the archive medium is stored
contiguously.

Bytes representing the contents of files (after the header record of
each file) are not translated in any way and are not constrained to
represent characters in any character set.  The @code{tar} format
does not distinguish text files from binary files, and no
translation of file contents is performed.

The @code{name}, @code{linkname}, @code{magic}, @code{uname}, and
@code{gname} are null-terminated character strings.  All other
fileds are zero-filled octal numbers in ASCII.  Each numeric field
of width @var{w} contains @var{w}@minus{} 2 digits, a space, and a null,
except @code{size}, and @code{mtime}, which do not contain the
trailing null.
```

The @code{name} field is the pathname of the file, with directory names (if any) preceding the file name, separated by slashes.

The @code{mode} field provides nine bits specifying file permissions and three bits to specify the Set UID, Set GID, and Save Text (``stick'') modes. Values for these bits are defined above. When special permissions are required to create a file with a given mode, and the user restoring files from the archive does not hold such permissions, the mode bit(s) specifying those special permissions are ignored. Modes which are not supported by the operating system restoring files from the archive will be ignored. Unsupported modes should be faked up when creating or updating an archive; e.g. the group permission could be copied from the @code{other} permission.

The @code{uid} and @code{gid} fields are the numeric user and group ID of the file owners, respectively. If the operating system does not support numeric user or group IDs, these fields should be ignored.

The @code{size} field is the size of the file in bytes; linked files are archived with this field specified as zero.
@xref{Extraction Options}; in particular the @samp{-G} option.@refill

The @code{mtime} field is the modification time of the file at the time it was archived. It is the ASCII representation of the octal value of the last time the file was modified, represented as an integer number of seconds since January 1, 1970, 00:00 Coordinated Universal Time.

The @code{chksum} field is the ASCII representation of the octal value of the simple sum of all bytes in the header record. Each 8-bit byte in the header is added to an unsigned integer, initialized to zero, the precision of which shall be no less than seventeen bits. When calculating the checksum, the @code{chksum} field is treated as if it were all blanks.

The @code{typeflag} field specifies the type of file archived. If a particular implementation does not recognize or permit the specified type, the file will be extracted as if it were a regular file. As this action occurs, @code{tar} issues a warning to the standard error.

@table @code
@item LF_NORMAL
@itemx LF_OLDNORMAL
These represent a regular file. In order to be compatible with older versions of @code{tar}, a @code{typeflag} value of @code{LF_OLDNORMAL} should be silently recognized as a regular file. New archives should be created using @code{LF_NORMAL}. Also, for backward compatibility, @code{tar} treats a regular file whose name ends with a slash as a directory.

@item LF_LINK
This represents a file linked to another file, of any type,
previously archived. Such files are identified in Unix by each file
having the same device and inode number. The linked-to
name is specified in the @code{linkname} field with a trailing null.

@item LF_SYMLINK
This represents a symbolic link to another file. The linked-to
name is specified in the @code{linkname} field with a trailing null.

@item LF_CHR
@itemx LF_BLK
These represent character special files and block special files
respectively. In this case the @code{devmajor} and @code{devminor}
fields will contain the major and minor device numbers
respectively. Operating systems may map the device specifications
to their own local specification, or may ignore the entry.

@item LF_DIR
This specifies a directory or sub-directory. The directory name in
the @code{name} field should end with a slash. On systems where
disk allocation is performed on a directory basis the @code{size}
field will contain the maximum number of bytes (which may be rounded
to the nearest disk block allocation unit) which the directory may
hold. A @code{size} field of zero indicates no such limiting.
Systems which do not support limiting in this manner should ignore
the @code{size} field.

@item LF_FIFO
This specifies a FIFO special file. Note that the archiving of a
FIFO file archives the existence of this file and not its contents.

@item LF_CONTIG
This specifies a contiguous file, which is the same as a normal
file except that, in operating systems which support it,
all its space is allocated contiguously on the disk. Operating
systems which do not allow contiguous allocation should silently treat
this type as a normal file.

@item 'A' @dots{}
@itemx 'Z'
These are reserved for custom implementations. Some of these are
used in the GNU modified format, as described below.
@end table

Other values are reserved for specification in future revisions of
the P1003 standard, and should not be used by any @code{tar} program.

The @code{magic} field indicates that this archive was output in the
P1003 archive format. If this field contains @code{TMAGIC}, the
@code{uname} and @code{gname} fields will contain the ASCII

representation of the owner and group of the file respectively. If found, the user and group ID represented by these names will be used rather than the values within the @code{uid} and @code{gid} fields.

@section GNU Extensions to the Archive Format
The GNU format uses additional file types to describe new types of files in an archive. These are listed below.

@table @code
@item LF_DUMPDIR
@itemx 'D'
This represents a directory and a list of files created by the @samp{-G} option. The @code{size} field gives the total size of the associated list of files. Each filename is preceded by either a @code{'Y'} (the file should be in this archive) or an @code{'N'} (The file is a directory, or is not stored in the archive). Each filename is terminated by a null. There is an additional null after the last filename.

@item LF_MULTIVOL
@itemx 'M'
This represents a file continued from another volume of a multi-volume archive created with the @samp{-M} option. The original type of the file is not given here. The @code{size} field gives the maximum size of this piece of the file (assuming the volume does not end before the file is written out). The @code{offset} field gives the offset from the beginning of the file where this part of the file begins. Thus @code{size} plus @code{offset} should equal the original size of the file.

@item LF_VOLHDR
@itemx 'V'
This file type is used to mark the volume header that was given with the @samp{-V} option when the archive was created. The @code{name} field contains the @code{name} given after the @samp{-V} option. The @code{size} field is zero. Only the first file in each volume of an archive should have this type.

@end table
EXTENSION:
OCCURENCES:
PROGRAMS:
REFERENCE:
SEE ALSO:
VALIDATION:
OFFSET   Count TYPE   Description
0000h   256 byte   Other header info ?
0100h   6 char   ID='ustar',0
EXTENSION:TAR
OCCURENCES:PC, Unix
PROGRAMS:TAR