NAME

tar — format of tape archive files

DESCRIPTION

The **tar** archive format collects any number of files, directories, and other file system objects (symbolic links, device nodes, etc.) into a single stream of bytes. The format was originally designed to be used with tape drives that operate with fixed-size blocks, but is widely used as a general packaging mechanism.

General Format

A **tar** archive consists of a series of 512-byte records. Each file system object requires a header record which stores basic metadata (pathname, owner, permissions, etc.) and zero or more records containing any file data. The end of the archive is indicated by two records consisting entirely of zero bytes.

For compatibility with tape drives that use fixed block sizes, programs that read or write tar files always read or write a fixed number of records with each I/O operation. These "blocks" are always a multiple of the record size. The maximum block size supported by early implementations was 10240 bytes or 20 records. This is still the default for most implementations although block sizes of 1MiB (2048 records) or larger are commonly used with modern high-speed tape drives. (Note: the terms "block" and "record" here are not entirely standard; this document follows the convention established by John Gilmore in documenting **pdtar**.)

Old-Style Archive Format

The original tar archive format has been extended many times to include additional information that various implementors found necessary. This section describes the variant implemented by the tar command included in Version 7 AT&T UNIX, which seems to be the earliest widely-used version of the tar program.

The header record for an old-style **tar** archive consists of the following:

```
struct header_old_tar {
        char name[100];
        char mode[8];
        char uid[8];
        char gid[8];
        char size[12];
        char mtime[12];
        char checksum[8];
        char linkflag[1];
        char linkname[100];
        char pad[255];
};
```

All unused bytes in the header record are filled with nulls.

name Pathname, stored as a null-terminated string. Early tar implementations only stored regular files (including hardlinks to those files). One common early convention used a trailing "/" character to indicate a directory name, allowing directory permissions and owner information to be archived and restored.

mode File mode, stored as an octal number in ASCII.

uid, *gid* User id and group id of owner, as octal numbers in ASCII.

size Size of file, as octal number in ASCII. For regular files only, this indicates the amount of data that follows the header. In particular, this field was ignored by early tar implementations when extracting hardlinks. Modern writers should always store a zero length for hardlink entries.

mtime Modification time of file, as an octal number in ASCII. This indicates the number of seconds since the start of the epoch, 00:00:00 UTC January 1, 1970. Note that negative values should be avoided here, as they are handled inconsistently.

checksum
Header checksum, stored as an octal number in ASCII. To compute the checksum, set the checksum field to all spaces, then sum all bytes in the header using unsigned arithmetic. This field should be stored as six octal digits followed by a null and a space character. Note that many early implementations of tar used signed arithmetic for the checksum field, which can cause interoperability problems when transferring archives between systems. Modern robust readers compute the checksum both ways and accept the header if either computation matches.

linkflag, *linkname*
In order to preserve hardlinks and conserve tape, a file with multiple links is only written to the archive the first time it is encountered. The next time it is encountered, the *linkflag* is set to an ASCII '1' and the *linkname* field holds the first name under which this file appears. (Note that regular files have a null value in the *linkflag* field.)

Early tar implementations varied in how they terminated these fields. The tar command in Version 7 AT&T UNIX used the following conventions (this is also documented in early BSD manpages): the pathname must be null-terminated; the mode, uid, and gid fields must end in a space and a null byte; the size and mtime fields must end in a space; the checksum is terminated by a null and a space. Early implementations filled the numeric fields with leading spaces. This seems to have been common practice until the IEEE Std 1003.1-1988 ("POSIX.1") standard was released. For best portability, modern implementations should fill the numeric fields with leading zeros.

Pre-POSIX Archives

An early draft of IEEE Std 1003.1-1988 ("POSIX.1") served as the basis for John Gilmore's **pdtar** program and many system implementations from the late 1980s and early 1990s. These archives generally follow the POSIX ustar format described below with the following variations:

- The magic value consists of the five characters "ustar" followed by a space. The version field contains a space character followed by a null.
- The numeric fields are generally filled with leading spaces (not leading zeros as recommended in the final standard).
- The prefix field is often not used, limiting pathnames to the 100 characters of old-style archives.

POSIX ustar Archives

IEEE Std 1003.1-1988 ("POSIX.1") defined a standard tar file format to be read and written by compliant implementations of `tar`(1). This format is often called the "ustar" format, after the magic value used in the header. (The name is an acronym for "Unix Standard TAR".) It extends the historic format with new fields:

```
struct header_posix_ustar {
        char name[100];
        char mode[8];
        char uid[8];
        char gid[8];
        char size[12];
        char mtime[12];
        char checksum[8];
        char typeflag[1];
        char linkname[100];
        char magic[6];
        char version[2];
        char uname[32];
```