# 什么是 tar

孔俊

# **Contents**

tar 是 *tape archive* 的简称，即"磁带归档"，最早出现在 1979 年的第 UNIX v7。磁带上的数据存储为可能不相邻的变长数据块，浪费了大量块与块之间的空间。向磁盘、网络传输数据时，传输一大块的效率远高于多个小块，因此程序员使用 tar 将磁带上的数据打包在一起，物理上存储在连续的、固定大小的块上以提高性能。

tar 打包出来的文件称为为 tarball。一般会使用某种压缩算法压缩 tarball，压缩后的 tarball 后缀一般为 `tar.<compress-algorithm>`，比如 gzip 压缩的 tarball 名为 `tar.gz`，zstd 压缩的 tarball 名为 `tar.zstd`。

# 1 场景

tar 被广泛使用，一个经典的场景是数据备份，将一个目录打包成 tarball 拷贝到别的机器并恢复。

tar 不仅被用于上述运维/系统管理领域，在容器中也有应用。容器镜像是分层存储的，其中每一层都是一个 tarball（可能被压缩）。

# 2 格式

最早的格式是 UNIX v7 格式，即 UNIX V7 中 tar 所使用的格式，，后来的各种 tar 格式都基于此格式拓展。

目前广泛使用的格式有 UStar、pax 和 GNU，其中 UStar 和 pax 是 POSIX 标准，GNU 格式未标准化但被广泛使用。

tarball 包含一系列文件对象，文件对象在物理上存储为一个或多个块 *(block)*，tarball 末尾由两个连续的内容为零（\0）的块标示。文件对象是一个 header 和 data 的组合、类似于 TCP 报文段。

最初的 UNIX V7 tar 不要求块末尾的填充（*padding*）为零，但现代实现一般都把填充内容设置为空字符\0。

大体上 tar 的格式是这样的，pax/gnu 等格式在此基础上有一些拓展，比如 pax 格式中将 *extend header data* 也存放在文件对象的 data 部分，所以 data 部分可能不仅仅是原始的文件数据。

## 2.1 v7

v7 格式 header 如下：

| Field offset | Field size | Field |
| --- | --- | --- |
| 0 | 100 | File name |
| 100 | 8 | File mode (octal) |

| Field offset | Field size | Field |
| --- | --- | --- |
| 108 | 8 | Owner's numeric user ID (octal) |
| 116 | 8 | Group's numeric user ID (octal) |
| 124 | 12 | File size in bytes (octal) |
| 136 | 12 | Last modification time in numeric Unix time format (octal) |
| 148 | 8 | Checksum for header record |
| 156 | 1 | Link indicator (file type) |
| 157 | 100 | Name of linked file |

link indicator 字段可以取以下值:

| Value | Meaning |
| --- | --- |
| '0' or (ASCII NUL) | Normal file |
| '1' | Hard link |
| '2' | Symbolic link |

tarball 被设计为一个文本文件，header 的各字段都是 ASCII 编码的。因此，对于一个原始文件都是文本的 tarball，那么它自身也一定是一个文本文件。

文件大小使用 ASCII 编码的八进制表示，并且字段最后一个字节为空，实际上只有 11 个字节用于编码，所以 v7 格式单个文件大小最大为 8GB（077777777 字节）。可见 v7 格式非常简单，不包含 xattr、group name、user name、MIME 类型、文件编码等关键信息，并且文件名长度也限制在 99 字节（不算末尾的\0）。

tar 对硬链接的处理有专门的规定，即将硬链接指向的文件写入到 **tarball** 中，其他硬链接作为 **hard link** 写入 **tarball**。GNU tar `--hard-dereference`将所有硬链接当成独立的内容相同的文件处理。

> If hard links exist, the target is the first occurrence of the hard link that is saved in the archive. Subsequent hard links refer to the first instance.

这个示例展示 tar 如何处理符号链接和硬链接。

```
 1  $ echo "TEST" > test
 2  $ for i in $(seq 1 3); do
 3  ln /tmp/test /tmp/test.link${i}
 4  done
 5  $ ls -li test*
 6  18451488 -rw-r--r-- 4 kongjun kongjun 5  9月 21 10:22 test
 7  18451488 -rw-r--r-- 4 kongjun kongjun 5  9月 21 10:22 test.link1
 8  18451488 -rw-r--r-- 4 kongjun kongjun 5  9月 21 10:22 test.link2
 9  18451488 -rw-r--r-- 4 kongjun kongjun 5  9月 21 10:22 test.link3
10  $ tar -cv -f hardlink.tar test*
11  test
12  test.link1
13  test.link2
14  test.link3
15   只写入一次文件
16  $ tar -tv -f hardlink.tar
17  -rw-r--r-- kongjun/kongjun   5 2022-09-21 10:22 test
18  hrw-r--r-- kongjun/kongjun   0 2022-09-21 10:22 test.link1 连接到 test
19  hrw-r--r-- kongjun/kongjun   0 2022-09-21 10:22 test.link2 连接到 test
20  hrw-r--r-- kongjun/kongjun   0 2022-09-21 10:22 test.link3 连接到 test
21  $ tar -cv --hard-dereference -f hardlink.tar test*
22  test
23  test.link1
24  test.link2
25  test.link3
26   所有硬链接当成内容完全相同的文件处理
27  $ tar -tv -f hardlink.tar
28  -rw-r--r-- kongjun/kongjun   5 2022-09-21 10:22 test
29  -rw-r--r-- kongjun/kongjun   5 2022-09-21 10:22 test.link1
30  -rw-r--r-- kongjun/kongjun   5 2022-09-21 10:22 test.link2
31  -rw-r--r-- kongjun/kongjun   5 2022-09-21 10:22 test.link3
```

## 2.2 UStar

UStar(Unix Standard TAR) 是 POSIX 标准化（POSIX.1-1988 and POSIX.1-2001）的格式，GNU tar 文档称为基础 tar 格式（*basic tar format*），通常现代 tar 都支持此格式。

UStar 在 v7 的基础上，拓展了 header，支持更多文件类型，添加了所有者名字，增大了文件名最大长度。

| Field offset | Field size | Field |
| --- | --- | --- |
| 0 | 156 | *(Several fields, same as in the old format)* |
| 156 | 1 | Type flag |
| 157 | 100 | *(Same field as in the old format)* |
| 257 | 6 | UStar indicator "ustar" then NUL |

| Field offset | Field size | Field |
|---|---|---|
| 263 | 2 | UStar version "00" |
| 265 | 32 | Owner user name |
| 297 | 32 | Owner group name |
| 329 | 8 | Device major number |
| 337 | 8 | Device minor number |
| 345 | 155 | Filename prefix |

v7 中的 link indicator 在 UStar 中拓展为 type flag，可以取以下值：

| Value | Meaning |
|---|---|
| '0' or (ASCII NUL) | Normal file |
| '1' | Hard link |
| '2' | Symbolic link |
| '3' | Character special |
| '4' | Block special |
| '5' | Directory |
| '6' | FIFO |
| '7' | Contiguous file |
| 'g' | Global extended header with meta data (POSIX.1-2001) |
| 'x' | Extended header with meta data for the next file in the archive (POSIX.1-2001) |
| 'A' – 'Z' | Vendor specific extensions (POSIX.1-1988) |
| All other values | Reserved for future standardization |

其中 continuous file 指此文件在存储在磁盘连续的块中，OS 一般不提供这样的接口，因此几乎所有 tar 实现都把忽略此类型，当成一般的文件处理。

x 和 g 类型用于 pax 格式，这样的 header 称为 pax header block，其数据是 pax extended header。

## 2.3 pax

前面的 v7、UStar 格式都有很多限制，比如文件名长度有限，无法记录 ACL 信息，无法记录文件 MIME 类型等信息。这些问题可以通过打标签统一解决，比如 ACL 在文件系统层面就是 xattr 而已，在 tar 中自然也可通过打相应标签解决。

pax(*portable archive interchange*) 格式最早由 Sun Microsystems 发明，由 POSIX.1-2001 标准化，GNU tar 文档称指为 POSIX 格式。

pax 被设计为，凡是支持 UStar 的 tar 实现，都可以处理 pax 格式。具体地说，pax 没有带来任何不兼容的新字段，只是为 UStar header/data 添加了新的语义。pax 分为 pax header block 和 pax extended header 两部分，pax header 是 type flag 为 x 或 g 的 UStar header，pax extended header 是拓展的元数据，作为 Ustar 数据存储。

pax 格式 tarball 布局如下：

| Header block | File |
| --- | --- |
| UStar Header [`typeflag=g`] | Global Extended Header. |
| Global Extended Header Data | Global Extended Header. |
| UStar Header [`typeflag=x`] | File 1: Extended Header is included. |
| Extended Header Data | File 1: Extended Header is included. |
| UStar Header [`typeflag=0`] | File 1: Extended Header is included. |
| Data for File 1 | File 1: Extended Header is included. |
| UStar Header [`typeflag=0`] | File 2: No Extended Header is included. |
| Data for File 2 | File 2: No Extended Header is included. |
| Block of binary zeros | End of Archive Indicator. |

可见，pax 格式的 tarball 完全可以当成 UStar 格式处理，只是会数去 pax 格式添加的许多信息而已。为了让只支持 UStar 格式的 tar 也能正常处理 pax 格式，pax 建议实现

1. 设置合法的文件名以便只支持 UStar 格式的 tar 能正确创建文件。

2. 文件名长度不不要超过 99 字节，否则无法被只支持 UStar 格式的 tar 正确处理。

pax header block 目前只有两种类型：

- x：表示该拓展头只作用于 tarball 中的下一个文件。

- g：表示该拓展头是全局的，作用于 tarball 中所有再其后面的文件。

pax extended header 是拓展的头，即拓展的元数据（标签），格式为`"%d %s=%s\n"`，`<length>`，`<keyword>`，`<value>`。其中`length`是包括末尾的换行符的八进制编码的长度（和 v7/UStar 保存一致），`keyword`和`value`必须使用Portable Filename Character Set中的字符。extended header 的字符编码为 UTF-8。

类型为 x 的 extended header 可以覆盖全局 extended header 中的`keyword`。`keyword`值为空时，删除该`keyword`；`keyword`值不为空时，增加或覆盖该`keyword`。extended header 中的字段即可以

- 记录额外元数据。比如 atime 等信息。

- 调整 UStar header option 的行为。比如 hdrcharset 可以修改 UStar header 中的文件名字符编码。

- 覆盖 UStar header option。比如 size 可以覆盖 UStar header 中记录的大小以支持大于 8G 的文件。

类似于 xattr(7)，pax extended header 中可使用的字段是预定义的。POSIX 标准规定的字段如下：

**atime**

The file access time for the following file(s), equivalent to the value of the *st_atime* member of the **stat** structure for a file, as described by the *stat*() function. The access time shall be restored if the process has appropriate privileges required to do so. The format of the *<value>* shall be as described in pax Extended Header File Times.

**charset**

The name of the character set used to encode the data in the following file(s). The entries in the following table are defined to refer to known standards; additional names may be agreed on between the originator and recipient.

|  | Formal Standard |
|---|---|
| ISO-IRΔ646Δ1990 | ISO/IEC 646:1990 |
| ISO-IRΔ8859Δ1Δ1998 | ISO/IEC 8859-1:1998 |
| ISO-IRΔ8859Δ2Δ1999 | ISO/IEC 8859-2:1999 |
| ISO-IRΔ8859Δ3Δ1999 | ISO/IEC 8859-3:1999 |
| ISO-IRΔ8859Δ4Δ1998 | ISO/IEC 8859-4:1998 |
| ISO-IRΔ8859Δ5Δ1999 | ISO/IEC 8859-5:1999 |
| ISO-IRΔ8859Δ6Δ1999 | ISO/IEC 8859-6:1999 |
| ISO-IRΔ8859Δ7Δ1987 | ISO/IEC 8859-7:1987 |

| | |
|---|---|
| ISO-IRΔ8859Δ8Δ1999 | ISO/IEC 8859-8:1999 |
| ISO-IRΔ8859Δ9Δ1999 | ISO/IEC 8859-9:1999 |
| ISO-IRΔ8859Δ10Δ1998 | ISO/IEC 8859-10:1998 |
| ISO-IRΔ8859Δ13Δ1998 | ISO/IEC 8859-13:1998 |
| ISO-IRΔ8859Δ14Δ1998 | ISO/IEC 8859-14:1998 |
| ISO-IRΔ8859Δ15Δ1999 | ISO/IEC 8859-15:1999 |
| ISO-IRΔ10646Δ2000 | ISO/IEC 10646:2000 |
| ISO-IRΔ10646Δ2000ΔUTF-8 | ISO/IEC 10646, UTF-8 encoding |
| BINARY | None. |

The encoding is included in an extended header for information only; when *pax* is used as described in POSIX.1-2017, it shall not translate the file data into any other encoding. The **BINARY** entry indicates unencoded binary data.

When used in **write** or **copy** mode, it is implementation-defined whether *pax* includes a **charset** extended header record for a file.

**comment**

A series of characters used as a comment. All characters in the *<value>* field shall be ignored by *pax*.

**gid**

The group ID of the group that owns the file, expressed as a decimal number using digits from the ISO/IEC 646:1991 standard. This record shall override the *gid* field in the following header block(s). When used in **write** or **copy** mode, *pax* shall include a *gid* extended header record for each file whose group ID is greater than 2097151(octal 7777777).

**gname**

The group of the file(s), formatted as a group name in the group database. This record shall override the *gid* and *gname* fields in the following header block(s), and any *gid* extended header record. When used in **read**, **copy**, or **list** mode, *pax* shall translate the name from the encoding in the header record to the character set appropriate for the group database on the receiving system. If any of the characters cannot be translated, and if neither the **-o invalid=UTF-8** option nor the **-o invalid=binary** option is specified, the results are implementation-defined. When used in **write** or **copy** mode, *pax* shall include a **gname** extended header record for each file whose group name cannot be represented entirely with the letters and digits of the portable character set.

**hdrcharset**

The name of the character set used to encode the value field of the **gname**, **linkpath**, **path**, and **uname** *pax* extended header records. The entries in the following table are defined to refer to known standards; additional names may be agreed between the originator and the recipient.

|  | Formal Standard |
| --- | --- |
| ISO-IRΔ10646Δ2000ΔUTF-8 | ISO/IEC 10646, UTF-8 encoding |
| BINARY | None. |

If no **hdrcharset** extended header record is specified, the default character set used to encode all values in extended header records shall be the ISO/IEC 10646-1:2000 standard UTF-8 encoding.

The **BINARY** entry indicates that all values recorded in extended headers for affected files are unencoded binary data from the underlying system.

**linkpath**

The pathname of a link being created to another file, of any type, previously archived. This record shall override the *linkname* field in the following **ustar** header block(s). The following **ustar** header block shall determine the type of link created. If *typeflag* of the following header block is 1, it shall be a hard link. If *typeflag* is 2, it shall be a symbolic link and the **linkpath** value shall be the contents of the symbolic link. The *pax* utility shall translate the name of the link (contents of the symbolic link) from the encoding in the header to the character set appropriate for the local file system. When used in **write** or **copy** mode, *pax* shall include a **linkpath** extended header record for each link whose pathname cannot be represented entirely with the members of the portable character set other than NUL.

**mtime**

The file modification time of the following file(s), equivalent to the value of the *st_mtime* member of the **stat** structure for a file, as described in the *stat*() function. This record shall override the *mtime* field in the following header block(s). The modification time shall be restored if the process has appropriate privileges required to do so. The format of the *<value>* shall be as described in pax Extended Header File Times.

**path**

The pathname of the following file(s). This record shall override the *name* and *prefix* fields in the following header block(s). The *pax* utility shall translate the pathname of the file from the encoding in the header to the character set appropriate for the local file system.

When used in **write** or **copy** mode, *pax* shall include a *path* extended header record for each file whose pathname cannot be represented entirely with the members of the portable character set other than NUL.

**realtime**.any

The keywords prefixed by "realtime." are reserved for future standardization.

**security**.any

The keywords prefixed by "security." are reserved for future standardization.

**size**

The size of the file in octets, expressed as a decimal number using digits from the ISO/IEC 646:1991 standard. This record shall override the *size* field in the following header block(s). When used in **write** or **copy** mode, *pax* shall include a *size* extended header record for each file with a size value greater than 8589934591 (octal 77777777777).

**uid**

The user ID of the file owner, expressed as a decimal number using digits from the ISO/IEC 646:1991 standard. This record shall override the *uid* field in the following header block(s). When used in **write** or **copy** mode, *pax* shall include a *uid* extended header record for each file whose owner ID is greater than 2097151 (octal 7777777).

**uname**

The owner of the following file(s), formatted as a user name in the user database. This record shall override the *uid* and *uname* fields in the following header block(s), and any *uid* extended header record. When used in **read**, **copy**, or **list** mode, *pax* shall translate the name from the encoding in the header record to the character set appropriate for the user database on the receiving system. If any of the characters cannot be translated, and if neither the **-o invalid=UTF-8** option nor the **-o invalid=binary** option is specified, the results are implementation-defined. When used in **write** or **copy** mode, *pax* shall include a **uname** extended header record for each file whose user name cannot be represented entirely with the letters and digits of the portable character set.

## 2.4 Old GNU

Old GNU 格式指 GNU tar 在 1990 年于 v1.09 引入的格式，该格式建立在 POSIX.1-1988 UStar 格式上，覆盖了一些 POSIX 要求的字段，造成兼容问题。因此，此格式已被弃用，GNU tar 文档中甚至没有详细介绍。

## 2.5 GNU

GNU 格式早于 pax 格式，基于早期 UStar 格式，解决了 UStar 格式不支持超过 8G 的大文件、长度超过 100 个字符的文件名等缺陷。

pax 格式标准化后，GNU 格式不再有存在的意义，GNU tar 文档声明将来会将默认格式迁移到 pax。

## 2.6 对比

| Attribute | USTAR | PAX | GNU |
|---|---|---|---|
| Name | 256B | unlimited | unlimited |
| Linkname | 100B | unlimited | unlimited |
| Size | uint33 | unlimited | uint89 |
| Mode | uint21 | uint21 | uint57 |
| Uid/Gid | uint21 | unlimited | uint57 |
| Uname/Gname | 32B | unlimited | 32B |
| ModTime | uint33 | unlimited | int89 |
| AccessTime | n/a | unlimited | int89 |
| ChangeTime | n/a | unlimited | int89 |
| Devmajor/Devminor | uint21 | uint21 | uint57 |
| string encoding | ASCII | UTF-8 | binary |
| sub-second times | no | yes | no |
| sparse files | no | yes | yes |

## 3 问题

### 3.1 **tar** 炸弹

- tarball 绝对路径或父级路径，解压后覆盖同名文件。

- 创建 tarball 前，攻击者创建指向敏感文件（如 /etc/passwd）的符号链接，创建 tarball 默认跟随符号链接，导致敏感文件被错误归档。

- 创建 tarball 时，攻击者将文件/目录修改为符号链接，比如`tar /home/usera`，攻击者将 /home/usera 设置为指向 /home/userb 的符号链接，导致 userb 目录被错误归档。

- 使用两个同名路径，第一个是符号链接，第二个是常规文件，解压后覆盖了符号链接指向的文件。

### 3.2 顺序访问

tar 最初是做磁带归档的软件，磁带不支持顺序访问，tar 自然也不支持。`tar`命令的`-u`(`--update`)并不是覆盖 tarball 中的文件，而是找到 tarball 的同名文件，比较最近修改时间，将更新的版本附加到末尾，解压后最后的文件是最终提取出来的版本。

下面的命令更新 connection.tar 中的文件 blues、fork、rock 和 classical，其中 blues 和 classical 两文件更新，附加到末尾。

```
 1  $ echo "classical" > classical
 2  $ echo "rock" > rock
 3  $ echo "folk" > folk
 4  $ echo "blues" > blues
 5  $ tar -cv -f collection.tar blues folk rock classical
 6  blues
 7  folk
 8  rock
 9  classical
10  $ tar -tv -f collection.tar
11  -rw-r--r-- kongjun/kongjun   6 2022-09-20 15:30 blues
12  -rw-r--r-- kongjun/kongjun   5 2022-09-20 15:30 folk
13  -rw-r--r-- kongjun/kongjun   5 2022-09-20 15:29 rock
14  -rw-r--r-- kongjun/kongjun  10 2022-09-20 15:29 classical
15  $ echo "bluesblues" > blues
16  $ echo "classicalclassical" > classical
17  $ tar --update -v -f collection.tar blues folk rock classical
18  blues
19  classical
20  $ tar -tv -f collection.tar
21  -rw-r--r-- kongjun/kongjun   6 2022-09-20 15:30 blues
22  -rw-r--r-- kongjun/kongjun   5 2022-09-20 15:30 folk
23  -rw-r--r-- kongjun/kongjun   5 2022-09-20 15:29 rock
24  -rw-r--r-- kongjun/kongjun  10 2022-09-20 15:29 classical
25  -rw-r--r-- kongjun/kongjun  11 2022-09-20 15:30 blues
26  -rw-r--r-- kongjun/kongjun  19 2022-09-20 15:30 classical
```

### 3.3 错误检验

tar 没有完善的错误校验机制。如果在传输过程中出错，难以依靠 header 中简单的 checksum 进行错误检测与恢复。

如果要验证数据完整性，可以在网络传输之前先使用 cksum 之类的命令计算 checksum，接受到 tarball 后比较。

恢复出错的数据几乎是不可能的，用户只能在了解 tar 格式的前提下，跳过错误的文件。

### 3.4 竞争条件

tar 只是将文件系统中的文件归档起来，不做任何高级的多余操作。归档过程中的文件修改、增删可能导致归档文件和原始数据不一致，甚至导致归档失败。

因此，必须确保归档过程中没有进程访问文件系统。备份系统中常用的办法是创建文件系统快照，然后归档此快照。

# **4** 实现

## **4.1 Go** 语言

Go 语言标准库包 archive/tar 实现了 tar，支持 ustar、pax 和 GNU 格式。Go 语言实现屏蔽了部分底层格式的差异，让用户聚焦于关键的 tar 属性。查看Header定义就可以发现这一点，定义如下：

```go
 1  type Header struct {
 2      // Typeflag is the type of header entry.
 3      // The zero value is automatically promoted to either TypeReg or
           TypeDir
 4      // depending on the presence of a trailing slash in Name.
 5      Typeflag byte
 6
 7      Name     string // Name of file entry
 8      Linkname string // Target name of link (valid for TypeLink or
           TypeSymlink)
 9
10      Size  int64  // Logical file size in bytes
11      Mode  int64  // Permission and mode bits
12      Uid   int    // User ID of owner
13      Gid   int    // Group ID of owner
14      Uname string // User name of owner
15      Gname string // Group name of owner
16
17      // If the Format is unspecified, then Writer.WriteHeader rounds
           ModTime
18      // to the nearest second and ignores the AccessTime and ChangeTime
           fields.
19      //
20      // To use AccessTime or ChangeTime, specify the Format as PAX or
           GNU.
21      // To use sub-second resolution, specify the Format as PAX.
22      ModTime    time.Time // Modification time
23      AccessTime time.Time // Access time (requires either PAX or GNU
           support)
24      ChangeTime time.Time // Change time (requires either PAX or GNU
           support)
25
26      Devmajor int64 // Major device number (valid for TypeChar or
           TypeBlock)
27      Devminor int64 // Minor device number (valid for TypeChar or
           TypeBlock)
28
29      // Xattrs stores extended attributes as PAX records under the
30      // "SCHILY.xattr." namespace.
31      //
32      // The following are semantically equivalent:
33      //  h.Xattrs[key] = value
34      //  h.PAXRecords["SCHILY.xattr."+key] = value
35      //
36      // When Writer.WriteHeader is called, the contents of Xattrs will
           take
37      // precedence over those in PAXRecords.
38      //
39      // Deprecated: Use PAXRecords instead.
40      Xattrs map[string]string
41
42      // PAXRecords is a map of PAX extended header records.
43      //
44      // User-defined records should have keys of the following form:
45      //  VENDOR.keyword
46      // Where VENDOR is some namespace in all uppercase, and keyword may
47      // not contain the '=' character (e.g., "GOLANG.pkg.version").
48      // The key and value should be non-empty UTF-8 strings.
49      //
50      // When Writer.WriteHeader is called, PAX records derived from the
51      // other fields in Header take precedence over PAXRecords.
```

## 4.2 GNU tar

GNU tar 是 GNU/Linux 上默认的 tar 实现，目前默认使用 GNU 格式，但文档声明将来会迁移到 pax 格式。

为了缓解安全问题，GNU tar 默认不解引用符号链接，创建归档时去除路径开头的 /。

# 5 思考

- tar 只做好归档，它不完美，但足够好。40 年后的今天，仍然活跃于容器镜像中。

- tar 格式从 v7 发展到 ustar，再到如今的 pax，展示了怎样在保持兼容的前提下拓展。

- 与其定死属性（attribute），不如考虑下打标签，打标签更易拓展。

# 6 参考

- GNU tar 1.34: 8 Controlling the Archive Format
- GNU tar 1.34: 4.2.3.1 How to Update an Archive Using –update
- pax - portable archive interchange
- IBM Document z/OS/2.3.0/pax - Interchange portable archives
- IBM Documentation tar
- BSD tar
- tar(1) - Linux manual page
- tar - Dereferencing hard links - Unix &Linux Stack Exchange
- tar (computing) - Wikipedia
- GNU tar 1.34: 10.2 Security
- GNU tar 1.34: 10.1 Reliability
- tar package - archive/tar - Go Packages