NAME

cpio — format of cpio archive files

DESCRIPTION

The **cpio** archive format collects any number of files, directories, and other file system objects (symbolic links, device nodes, etc.) into a single stream of bytes.

General Format

Each file system object in a **cpio** archive comprises a header record with basic numeric metadata followed by the full pathname of the entry and the file data. The header record stores a series of integer values that generally follow the fields in *struct stat*. (See `stat`(2) for details.) The variants differ primarily in how they store those integers (binary, octal, or hexadecimal). The header is followed by the pathname of the entry (the length of the pathname is stored in the header) and any file data. The end of the archive is indicated by a special record with the pathname "TRAILER!!!".

PWB format

XXX Any documentation of the original PWB/UNIX 1.0 format? XXX

Old Binary Format

The old binary **cpio** format stores numbers as 2-byte and 4-byte binary values. Each entry begins with a header in the following format:

```
struct header_old_cpio {
        unsigned short   c_magic;
        unsigned short   c_dev;
        unsigned short   c_ino;
        unsigned short   c_mode;
        unsigned short   c_uid;
        unsigned short   c_gid;
        unsigned short   c_nlink;
        unsigned short   c_rdev;
       unsigned short   c_mtime[2];
        unsigned short   c_namesize;
       unsigned short   c_filesize[2];
};
```

The *unsigned short* fields here are 16-bit integer values; the *unsigned int* fields are 32-bit integer values. The fields are as follows

magic The integer value octal 070707. This value can be used to determine whether this archive is written with little-endian or big-endian integers.

dev, *ino* The device and inode numbers from the disk. These are used by programs that read **cpio** archives to determine when two entries refer to the same file. Programs that synthesize **cpio** archives should be careful to set these to distinct values for each entry.

mode The mode specifies both the regular permissions and the file type. It consists of several bit fields as follows:

0170000	This masks the file type bits.
0140000	File type value for sockets.
0120000	File type value for symbolic links. For symbolic links, the link body is stored as file data.
0100000	File type value for regular files.
0060000	File type value for block special devices.
0040000	File type value for directories.
0020000	File type value for character special devices.
0010000	File type value for named pipes or FIFOs.

0004000 SUID bit.
0002000 SGID bit.
0001000 Sticky bit. On some systems, this modifies the behavior of executables and/or directories.
0000777 The lower 9 bits specify read/write/execute permissions for world, group, and user following standard POSIX conventions.

uid, *gid* The numeric user id and group id of the owner.

nlink The number of links to this file. Directories always have a value of at least two here. Note that hardlinked files include file data with every copy in the archive.

rdev For block special and character special entries, this field contains the associated device number. For all other entry types, it should be set to zero by writers and ignored by readers.

mtime Modification time of the file, indicated as the number of seconds since the start of the epoch, 00:00:00 UTC January 1, 1970. The four-byte integer is stored with the most-significant 16 bits first followed by the least-significant 16 bits. Each of the two 16 bit values are stored in machine-native byte order.

namesize
The number of bytes in the pathname that follows the header. This count includes the trailing NUL byte.

filesize The size of the file. Note that this archive format is limited to four gigabyte file sizes. See *mtime* above for a description of the storage of four-byte integers.

The pathname immediately follows the fixed header. If the **namesize** is odd, an additional NUL byte is added after the pathname. The file data is then appended, padded with NUL bytes to an even length.

Hardlinked files are not given special treatment; the full file contents are included with each copy of the file.

Portable ASCII Format

Version 2 of the Single UNIX Specification (“SUSv2”) standardized an ASCII variant that is portable across all platforms. It is commonly known as the “old character” format or as the “odc” format. It stores the same numeric fields as the old binary format, but represents them as 6-character or 11-character octal values.

```
struct cpio_odc_header {
        char    c_magic[6];
        char    c_dev[6];
        char    c_ino[6];
        char    c_mode[6];
        char    c_uid[6];
        char    c_gid[6];
        char    c_nlink[6];
        char    c_rdev[6];
        char    c_mtime[11];
        char    c_namesize[6];
        char    c_filesize[11];
};
```

The fields are identical to those in the old binary format. The name and file body follow the fixed header. Unlike the old binary format, there is no additional padding after the pathname or file contents. If the files being archived are themselves entirely ASCII, then the resulting archive will be entirely ASCII, except for the NUL byte that terminates the name field.

New ASCII Format

The "new" ASCII format uses 8-byte hexadecimal fields for all numbers and separates device numbers into separate fields for major and minor numbers.

```
struct cpio_newc_header {
        char    c_magic[6];
        char    c_ino[8];
        char    c_mode[8];
        char    c_uid[8];
        char    c_gid[8];
        char    c_nlink[8];
        char    c_mtime[8];
        char    c_filesize[8];
        char    c_devmajor[8];
        char    c_devminor[8];
        char    c_rdevmajor[8];
        char    c_rdevminor[8];
        char    c_namesize[8];
        char    c_check[8];
};
```

Except as specified below, the fields here match those specified for the old binary format above.

magic The string “070701”.

check This field is always set to zero by writers and ignored by readers. See the next section for more details.

The pathname is followed by NUL bytes so that the total size of the fixed header plus pathname is a multiple of four. Likewise, the file data is padded to a multiple of four bytes. Note that this format supports only 4 gigabyte files (unlike the older ASCII format, which supports 8 gigabyte files).

In this format, hardlinked files are handled by setting the filesize to zero for each entry except the last one that appears in the archive.

New CRC Format

The CRC format is identical to the new ASCII format described in the previous section except that the magic field is set to “070702” and the *check* field is set to the sum of all bytes in the file data. This sum is computed treating all bytes as unsigned values and using unsigned arithmetic. Only the least-significant 32 bits of the sum are stored.

HP variants

The **cpio** implementation distributed with HPUX used XXXX but stored device numbers differently XXX.

Other Extensions and Variants

Sun Solaris uses additional file types to store extended file data, including ACLs and extended attributes, as special entries in cpio archives.

XXX Others? XXX

SEE ALSO

cpio(1), tar(5)

STANDARDS

The **cpio** utility is no longer a part of POSIX or the Single Unix Standard. It last appeared in Version 2 of the Single UNIX Specification ("SUSv2"). It has been supplanted in subsequent standards by pax(1). The portable ASCII format is currently part of the specification for the pax(1) utility.

HISTORY

The original cpio utility was written by Dick Haight while working in AT&T's Unix Support Group. It appeared in 1977 as part of PWB/UNIX 1.0, the "Programmer's Work Bench" derived from Version 6 AT&T UNIX that was used internally at AT&T. Both the old binary and old character formats were in use by 1980, according to the System III source released by SCO under their "Ancient Unix" license. The character format was adopted as part of IEEE Std 1003.1-1988 ("POSIX.1"). XXX when did "newc" appear? Who invented it? When did HP come out with their variant? When did Sun introduce ACLs and extended attributes? XXX

BUGS

The "CRC" format is mis-named, as it uses a simple checksum and not a cyclic redundancy check.

The old binary format is limited to 16 bits for user id, group id, device, and inode numbers. It is limited to 4 gigabyte file sizes.

The old ASCII format is limited to 18 bits for the user id, group id, device, and inode numbers. It is limited to 8 gigabyte file sizes.

The new ASCII format is limited to 4 gigabyte file sizes.

None of the cpio formats store user or group names, which are essential when moving files between systems with dissimilar user or group numbering.

Especially when writing older cpio variants, it may be necessary to map actual device/inode values to synthesized values that fit the available fields. With very large filesystems, this may be necessary even for the newer formats.