NAME

cpio — copy files to and from archives

SYNOPSIS

cpio **-i** [*options*] [*pattern ...*] [< *archive*]
cpio **-o** [*options*] < *name-list* [> *archive*]
cpio **-p** [*options*] *dest-dir* < *name-list*

DESCRIPTION

cpio copies files between archives and directories. This implementation can extract from tar, pax, cpio, zip, jar, ar, and ISO 9660 cdrom images and can create tar, pax, cpio, ar, and shar archives.

The first option to **cpio** is a mode indicator from the following list:

-i Input. Read an archive from standard input (unless overridden) and extract the contents to disk or (if the **-t** option is specified) list the contents to standard output. If one or more file patterns are specified, only files matching one of the patterns will be extracted.

-o Output. Read a list of filenames from standard input and produce a new archive on standard output (unless overridden) containing the specified items.

-p Pass-through. Read a list of filenames from standard input and copy the files to the specified directory.

OPTIONS

Unless specifically stated otherwise, options are applicable in all operating modes.

-0, **--null**
Read filenames separated by NUL characters instead of newlines. This is necessary if any of the filenames being read might contain newlines.

-A (o mode only) Append to the specified archive. (Not yet implemented.)

-a (o and p modes) Reset access times on files after they are read.

-B (o mode only) Block output to records of 5120 bytes.

-C *size*
(o mode only) Block output to records of *size* bytes.

-c (o mode only) Use the old POSIX portable character format. Equivalent to **--format** *odc*.

-d, **--make-directories**
(i and p modes) Create directories as necessary.

-E *file*
(i mode only) Read list of file name patterns from *file* to list and extract.

-F *file*, **--file** *file*
Read archive from or write archive to *file*.

-f *pattern*
(i mode only) Ignore files that match *pattern*.

-H *format*, **--format** *format*
(o mode only) Produce the output archive in the specified format. Supported formats include:

cpio Synonym for *odc*.
newc The SVR4 portable cpio format.

odc The old POSIX.1 portable octet-oriented cpio format.
pax The POSIX.1 pax format, an extension of the ustar format.
ustar The POSIX.1 tar format.

The default format is *odc*. See `libarchive-formats`(5) for more complete information about the formats currently supported by the underlying `libarchive`(3) library.

-h, **--help**
Print usage information.

-I *file*
Read archive from *file*.

-i, **--extract**
Input mode. See above for description.

--insecure
(i and p mode only) Disable security checks during extraction or copying. This allows extraction via symbolic links, absolute paths, and path names containing '..' in the name.

-J, **--xz**
(o mode only) Compress the file with xz-compatible compression before writing it. In input mode, this option is ignored; xz compression is recognized automatically on input.

-j Synonym for **-y**.

-L (o and p modes) All symbolic links will be followed. Normally, symbolic links are archived and copied as symbolic links. With this option, the target of the link will be archived or copied instead.

-l, **--link**
(p mode only) Create links from the target directory to the original files, instead of copying.

--lrzip
(o mode only) Compress the resulting archive with `lrzip`(1). In input mode, this option is ignored.

--lz4 (o mode only) Compress the archive with lz4-compatible compression before writing it. In input mode, this option is ignored; lz4 compression is recognized automatically on input.

--lzma
(o mode only) Compress the file with lzma-compatible compression before writing it. In input mode, this option is ignored; lzma compression is recognized automatically on input.

--lzop
(o mode only) Compress the resulting archive with `lzop`(1). In input mode, this option is ignored.

--passphrase *passphrase*
The `passphrase` is used to extract or create an encrypted archive. Currently, zip is only a format that **cpio** can handle encrypted archives. You shouldn't use this option unless you realize how insecure use of this option is.

-m, **--preserve-modification-time**
(i and p modes) Set file modification time on created files to match those in the source.

-n, **--numeric-uid-gid**
(i mode, only with **-t**) Display numeric uid and gid. By default, **cpio** displays the user and group names when they are provided in the archive, or looks up the user and group names in the system password database.

--no-preserve-owner
(i mode only) Do not attempt to restore file ownership. This is the default when run by non-root users.

-O *file*
Write archive to *file*.

-o, **--create**
Output mode. See above for description.

-p, **--pass-through**
Pass-through mode. See above for description.

--preserve-owner
(i mode only) Restore file ownership. This is the default when run by the root user.

--quiet
Suppress unnecessary messages.

-R [user][:][group], **--owner** [user][:][group]
Set the owner and/or group on files in the output. If group is specified with no user (for example, **-R** *:wheel*) then the group will be set but not the user. If the user is specified with a trailing colon and no group (for example, **-R** *root:*) then the group will be set to the user's default group. If the user is specified with no trailing colon, then the user will be set but not the group. In **-i** and **-p** modes, this option can only be used by the super-user. (For compatibility, a period can be used in place of the colon.)

-r
(All modes.) Rename files interactively. For each file, a prompt is written to `/dev/tty` containing the name of the file and a line is read from `/dev/tty`. If the line read is blank, the file is skipped. If the line contains a single period, the file is processed normally. Otherwise, the line is taken to be the new name of the file.

-t, **--list**
(i mode only) List the contents of the archive to stdout; do not restore the contents to disk.

-u, **--unconditional**
(i and p modes) Unconditionally overwrite existing files. Ordinarily, an older file will not overwrite a newer file on disk.

-V, **--dot**
Print a dot to stderr for each file as it is processed. Superseded by **-v**.

-v, **--verbose**
Print the name of each file to stderr as it is processed. With **-t**, provide a detailed listing of each file.

--version
Print the program version information and exit.

-y
(o mode only) Compress the archive with bzip2-compatible compression before writing it. In input mode, this option is ignored; bzip2 compression is recognized automatically on input.

-Z
(o mode only) Compress the archive with compress-compatible compression before writing it. In input mode, this option is ignored; compression is recognized automatically on input.

-z
(o mode only) Compress the archive with gzip-compatible compression before writing it. In input mode, this option is ignored; gzip compression is recognized automatically on input.

EXIT STATUS

The **cpio** utility exits 0 on success, and >0 if an error occurs.

ENVIRONMENT

The following environment variables affect the execution of **cpio**:

`LANG` The locale to use. See `environ`(7) for more information.

`TZ` The timezone to use when displaying dates. See `environ`(7) for more information.

EXAMPLES

The **cpio** command is traditionally used to copy file hierarchies in conjunction with the `find`(1) command. The first example here simply copies all files from `src` to `dest`:

```
find src | cpio -pmud dest
```

By carefully selecting options to the `find`(1) command and combining it with other standard utilities, it is possible to exercise very fine control over which files are copied. This next example copies files from `src` to `dest` that are more than 2 days old and whose names match a particular pattern:

```
find src -mtime +2 | grep foo[bar] | cpio -pdmu dest
```

This example copies files from `src` to `dest` that are more than 2 days old and which contain the word "foobar":

```
find src -mtime +2 | xargs grep -l foobar | cpio -pdmu dest
```

COMPATIBILITY

The mode options i, o, and p and the options a, B, c, d, f, l, m, r, t, u, and v comply with SUSv2.

The old POSIX.1 standard specified that only **-i**, **-o**, and **-p** were interpreted as command-line options. Each took a single argument of a list of modifier characters. For example, the standard syntax allows **-imu** but does not support **-miu** or **-i -m -u**, since *m* and *u* are only modifiers to **-i**, they are not command-line options in their own right. The syntax supported by this implementation is backwards-compatible with the standard. For best compatibility, scripts should limit themselves to the standard syntax.

SEE ALSO

`bzip2`(1), `tar`(1), `gzip`(1), `mt`(1), `pax`(1), `libarchive`(3), `cpio`(5), `libarchive-formats`(5), `tar`(5)

STANDARDS

There is no current POSIX standard for the cpio command; it appeared in ISO/IEC 9945-1:1996 ("POSIX.1") but was dropped from IEEE Std 1003.1-2001 ("POSIX.1").

The cpio, ustar, and pax interchange file formats are defined by IEEE Std 1003.1-2001 ("POSIX.1") for the pax command.

HISTORY

The original **cpio** and **find** utilities were written by Dick Haight while working in AT&T's Unix Support Group. They first appeared in 1977 in PWB/UNIX 1.0, the "Programmer's Work Bench" system developed for use within AT&T. They were first released outside of AT&T as part of System III Unix in 1981. As a result, **cpio** actually predates **tar**, even though it was not well-known outside of AT&T until some time later.

This is a complete re-implementation based on the `libarchive`(3) library.

BUGS

The cpio archive format has several basic limitations: It does not store user and group names, only numbers. As a result, it cannot be reliably used to transfer files between systems with dissimilar user and group numbering. Older cpio formats limit the user and group numbers to 16 or 18 bits, which is insufficient for modern systems. The cpio archive formats cannot support files over 4 gigabytes, except for the “odc” variant, which can support files up to 8 gigabytes.