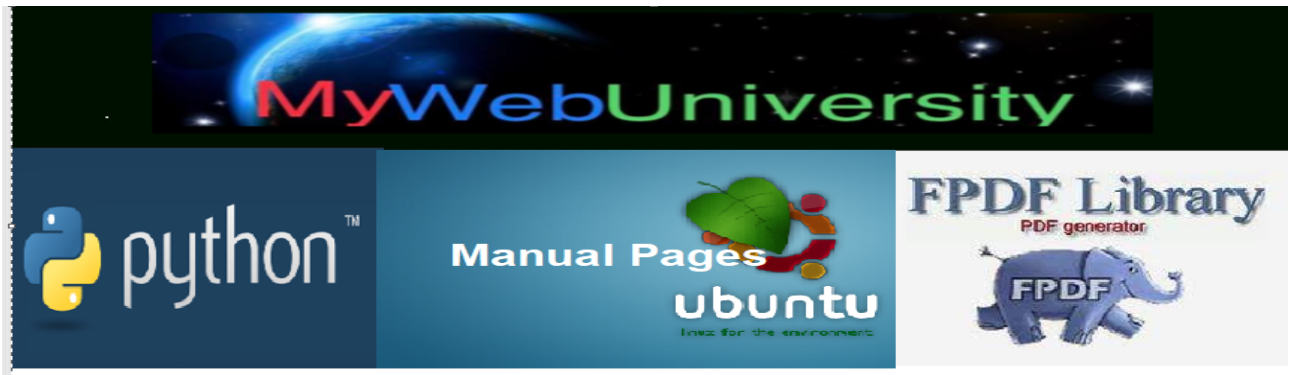

Full credit is given to the above companies including the OS that this PDF file was generated!

Linux Ubuntu 26.04 Manual Pages on command '3cpio.1'

$ man 3cpio.1

3CPIO(1) 3cpio 3CPIO(1)

NAME

3cpio - manage initrd cpio archives

SYNOPSIS

3cpio --count ARCHIVE

3cpio {-c|--create} [-v|--debug] [-C DIR] [--data-align ALIGNMENT] [ARCHIVE] < manifest

3cpio {-e|--examine} [--raw] ARCHIVE

3cpio {-t|--list} [-v|--debug] [-P LIST] ARCHIVE [pattern...]

3cpio {-x|--extract} [-v|--debug] [-C DIR] [-P LIST] [-p] [-s NAME] [--to-stdout] [--force]

[-F] ARCHIVE [-F|--file ARCHIVE...] [pattern...]

3cpio {-V|--version}

3cpio {-h|--help}

DESCRIPTION

3cpio is a tool to manage initramfs cpio files for the Linux kernel. The Linux kernel?s

initramfs buffer format

<https://www.kernel.org/doc/html/latest/driver-api/early-userspace/buffer-format.html> is

based around the newc or crc cpio formats. Multiple cpio archives can be concatenated and

the last archive can be compressed. Different compression algorithms can be used depending

on what support was compiled into the Linux kernel. 3cpio is tailored to initramfs cpio

files and will not gain support for other cpio formats.

Following compression formats are supported: bzip2, gzip, lz4, lzma, lzop, xz, zstd.

MODES

--count ARCHIVE

Print the number of concatenated cpio archives.

-c, --create [ARCHIVE]

Create a new cpio archive. Read the manifest from the standard input. See the MANIFEST section for the description of the manifest format. Write the cpio archive to standard output or to the specified ARCHIVE file if provided. The permission of the ARCHIVE file will be determined by the permission of the input files (to avoid leaking sensitive information).

-e, --examine ARCHIVE

List the offsets of the cpio archives and their compression in a formatted table. SI prefixes are used for the size unit (e. g. 1 kB = 1000 bytes). Do not rely on the output to have a specific layout. Use the --raw option for a machine-readable output. Then each line will contain these five values tab-separated:

Start

The beginning of the cpio archive in bytes.

End

The end of the cpio archive in bytes.

Size

The on-disk size of the cpio archive in bytes (end - start).

Compression

Compression of the cpio archive (cpio in case it is not compressed).

Extracted size

Size of the files inside the cpio archive in bytes.

-t, --list ARCHIVE [pattern...]

List the contents of the cpio archives. By default only the file names are printed. If --verbose is specified, the long listing format is used (similar to ls --long). If --debug is specified, the inode is printed in addition to the long format. If one or more patterns are supplied, list only file names matching at least one of those patterns. These patterns are shell wildcard patterns (see glob(7)).

-x, --extract [-F|--file] ARCHIVE [-F|--file ARCHIVE] [pattern...]

Extract cpio archives. Multiple archives can be specified with -F. The option -F can be omitted for the first archive. If one or more patterns are supplied, extract only files matching at least one of those patterns. These patterns are shell wildcard patterns (see glob(7)).

-V, --version

Print version number.

-h, --help

Print help message.

OPTIONS

--data-align ALIGNMENT

When creating a cpio archive, pad the cpio metadata to align the file data on ALIGNMENT in bytes. This option is useful to reflink cpio file data on file systems that support reflinks. This padding/alignment will be skipped for files that are smaller than ALIGNMENT, files where the padded namesize field would exceed PATH_MAX, and for cpio archives that are compressed. ALIGNMENT must be a multiple of 4 bytes. This option is only taken into account in the --create mode.

The resulting cpio archive may be highly fragmented, which can lead to performance degradation when reading/extracting the image from devices with slow random IO (e.g. spinning disk).

Note: Using this option will "bend" the cpio newc spec a bit to inject zeros after the filename to provide data segment alignment. These zeros are accounted for in the namesize, but some applications may only expect a single zero-terminator (and 4 byte alignment). GNU cpio and Linux initramfs handle this fine as long as PATH_MAX is not exceeded.

The following command can be used to determine the optimal transfer size of the file system (where $path is the path the cpio archive will be written to):

```
stat --file-system -c "%s" -- "$path"
```

-C DIR, --directory=DIR

Change directory before performing any operation, but after opening the ARCHIVE. This option is only taken into account in the --extract mode.

--make-directories: Create leading directories where needed. This option is only taken into account in the --extract mode.

-P LIST, --parts LIST

Only operate on the cpio archives that matches LIST. This option is only taken into account in the --list and --extract modes. LIST is made up of one range, or many ranges separated by commas. Each range is one of:

N

N?th cpio archive, counted from 1

N-

from N?th (included) to last cpio archive

N-M

from N?th to M?th (included) cpio archive

-M

from first to M?th (included) cpio archive

-p, --preserve-permissions

Set permissions of extracted files to those recorded in the archive (default for superuser). This option is only taken into account in the --extract mode.

--raw

Use a machine-readable output format. This format is designed for easy parsing and is intended for being used in scripts. This option is only taken into account in the --examine mode.

-s NAME, --subdir=NAME

Extract the cpio archives into separate sub-directories (using the given NAME plus an incrementing number). This option is only taken into account in the --extract mode.

--to-stdout

Extract files to standard output. Only the content of the files will be written to stdout. Directories and symlinks will be ignored. This option is only taken into account in the --extract mode.

-v, --verbose

Verbose output. This option is only taken into account in the --extract and --list modes.

--debug

Debug output. This option is only taken into account in the --extract and --list modes.

--force

Force overwriting existing files. This option is only taken into account in the --extract mode.

MANIFEST

When generating initrd cpio archives, following manifest format will be used. The manifest is a text format that is parsed line by line.

If the line starts with #cpio it is interpreted as section marker to start a new cpio. A

compression may be specified by adding a colon followed by the compression format and an optional compression level. Example for a Zstandard-compressed cpio with compression level 9:

#cpio: zstd -9

All lines starting with # excluding #cpio (see above) will be treated as comments and will be ignored.

Each element in the line is separated by a tab and is expected to be one of the following file types:

<location> <name> file <mode> <uid> <gid> <mtime> <filesize>

<location> <name> dir <mode> <uid> <gid> <mtime>

<location> <name> block <mode> <uid> <gid> <mtime> <major> <minor>

<location> <name> char <mode> <uid> <gid> <mtime> <major> <minor>

<location> <name> link <mode> <uid> <gid> <mtime> <target>

<location> <name> fifo <mode> <uid> <gid> <mtime>

<location> <name> sock <mode> <uid> <gid> <mtime>

fifo is also known as named pipe (see fifo(7)).

In case an element is empty or equal to - it is treated as not specified and it is derived from the input file.

<location>

Path of the input file. It can be left unspecified in case all other needed fields are specified (and the file is otherwise empty). Limitation: The path must not start with #, be equal to -, or contain tabs.

<name>

Path of the file inside the cpio. If the name is left unspecified it will be derived from <location>. Limitation: The path must not be equal to - or contain tabs.

<mode>

File mode specified in octal.

<uid>

User ID (owner) of the file specified in decimal.

<gid>

Group ID of the file specified in decimal.

<mtime>

Modification time of the file specified as seconds since the Epoch (1970-01-01 00:00

UTC). The specified time might be clamped by the time set in the SOURCE_DATE_EPOCH environment variable.

<filesize>

Size of the input file in bytes. 3cpio will fail in case the input file is smaller than the provided file size.

<major>

Major block/character device number in decimal.

<minor>

Minor block/character device number in decimal.

<target>

Target of the symbolic link. Limitation: The target path must not be equal to - or contain tabs.

Limitations: Files cannot start with # (will be treated as comment), be equal to - (will be treated as not specified), or contain tabs (will be split by tabs). These limitations of the manifest file are not expected to cause problems in practice.

ENVIRONMENT VARIABLES

SOURCE_DATE_EPOCH

This environment variable will be taken into account when creating cpio archive. All modification times that are newer than the time specified in "SOURCE_DATE_EPOCH" will be clamped. Compressors will run with only one thread in case their multithreading implementation is not reproducible. The created cpio archive will be reproducible across multiple runs.

EXIT STATUS

0

Success.

1

Failure.

2

Failure during command line argument parsing.

EXAMPLES

List the number of cpio archives that an initramfs file contains:

```
$ 3cpio --count /boot/initrd.img
4
```

Examine the content of the initramfs cpio on an Ubuntu 24.04 system:

```
$ 3cpio --examine /boot/initrd.img
Start     End       Size      Compr.   Extracted
0 B       148 kB    148 kB    cpio     147 kB
148 kB    13.3 MB   13.1 MB   cpio     13.1 MB
13.3 MB   55.2 MB   41.9 MB   cpio     41.7 MB
55.2 MB   62.0 MB   6.74 MB   zstd     15.6 MB
```

There is also a machine-readable output format available:

```
$ 3cpio --examine --raw /boot/initrd.img
0       148480       148480       cpio       147350
148480       13275136       13126656       cpio       13125632
13275136       55215104       41939968       cpio       41692226
55215104       61956920       6741816       zstd       15616306
```

This initramfs cpio consists of three uncompressed cpio archives followed by a Zstandard-compressed cpio archive.

List the content of the initramfs cpio on an Ubuntu 24.04 system:

```
$ 3cpio --list /boot/initrd.img
.
kernel
kernel/x86
kernel/x86/microcode
kernel/x86/microcode/AuthenticAMD.bin
kernel
kernel/x86
kernel/x86/microcode
kernel/x86/microcode/.enuineIntel.align.0123456789abc
kernel/x86/microcode/GenuineIntel.bin
.
usr
usr/lib
usr/lib/firmware
usr/lib/firmware/3com
usr/lib/firmware/3com/typhoon.bin.zst
```

```
[...]
```

The first cpio contains only the AMD microcode. The second cpio contains only the Intel microcode. The third cpio contains firmware files and kernel modules.

Extract the content of the initramfs cpio to the initrd subdirectory on an Ubuntu 24.04 system:

```
$ 3cpio --extract -C initrd /boot/initrd.img
$ ls initrd
bin   cryptroot  init    lib    lib.usr-is-merged  run   scripts  var
conf  etc        kernel  lib64  libx32             sbin  usr
```

Create a cpio archive similar to the other cpio tools using the find command:

```
$ cd inputdir && find . | sort | 3cpio --create ../example.cpio
```

Due to its manifest file format support, 3cpio can create cpio archives without the need of copying files into a temporary directory first. Example for creating an early microcode cpio image directly using the system installed files:

```
$ cat manifest
-       kernel        dir        755        0        0        1751654557
-       kernel/x86        dir        755        0        0        1752011622
/usr/lib/firmware/amd-ucode        kernel/x86/microcode
/usr/lib/firmware/amd-ucode/microcode_amd_fam19h.bin        kernel/x86/microcode/AuthenticAMD.bin
$ 3cpio --create amd-ucode.img < manifest
$ 3cpio --list --verbose amd-ucode.img
drwxr-xr-x   2 root     root            0 Jul  4 20:42 kernel
drwxr-xr-x   2 root     root            0 Jul  8 23:53 kernel/x86
drwxr-xr-x   2 root     root            0 Jun 10 10:51 kernel/x86/microcode
-rw-r--r--   1 root     root       100684 Mar 23 22:42 kernel/x86/microcode/AuthenticAMD.bin
```

Example for creating an initrd image containing of an uncompressed early microcode cpio followed by a Zstandard-compressed cpio:

```
$ cat manifest
#cpio
-       kernel        dir        755        0        0        1751654557
-       kernel/x86        dir        755        0        0        1752011622
/usr/lib/firmware/amd-ucode        kernel/x86/microcode
/usr/lib/firmware/amd-ucode/microcode_amd_fam19h.bin        kernel/x86/microcode/AuthenticAMD.bin
```

```
#cpio: zstd -9
/
/bin
/usr
/usr/bin
/usr/bin/bash
# This is a comment. Leaving the remaining files as task for the reader.
$ 3cpio --create initrd.img < manifest
$ 3cpio --examine initrd.img
Start     End       Size      Compr.   Extracted
0 B       101 kB    101 kB    cpio     101 kB
101 kB    786 kB    685 kB    zstd     1.45 MB
$ 3cpio --list --verbose initrd.img
drwxr-xr-x   2 root     root            0 Jul  4 20:42 kernel
drwxr-xr-x   2 root     root            0 Jul  8 23:53 kernel/x86
drwxr-xr-x   2 root     root            0 Jun 10 10:51 kernel/x86/microcode
-rw-r--r--   1 root     root       100684 Mar 23 22:42 kernel/x86/microcode/AuthenticAMD.bin
drwxr-xr-x   2 root     root            0 Jun  5 14:11 .
lrwxrwxrwx   1 root     root            7 Mar 20  2022 bin -> usr/bin
drwxr-xr-x   2 root     root            0 Apr 20  2023 usr
drwxr-xr-x   2 root     root            0 Jul  9 09:56 usr/bin
-rwxr-xr-x   1 root     root      1446024 Mar 31  2024 usr/bin/bash
```

SEE ALSO

bsdcpio(1), cpio(1), lsinitramfs(8), lsinitrd(1)

COPYING

AUTHOR

Benjamin Drung