initramfs buffer format

Al Viro, H. Peter Anvin

Last revision: 2002-01-13

Starting with kernel 2.5.x, the old "initial ramdisk" protocol is getting {replaced/complemented} with the new "initial ramfs" (initramfs) protocol. The initramfs contents is passed using the same memory buffer protocol used by the initrd protocol, but the contents is different. The initramfs buffer contains an archive which is expanded into a ramfs filesystem; this document details the format of the initramfs buffer format.

The initramfs buffer format is based around the "newc" or "crc" CPIO formats, and can be created with the cpio(1) utility. The cpio archive can be compressed using gzip(1). One valid version of an initramfs buffer is thus a single .cpio.gz file.

The full format of the initramfs buffer is defined by the following grammar, where:

```
*       is used to indicate "0 or more occurrences of"
(|)     indicates alternatives
+       indicates concatenation
GZIP()  indicates the gzip(1) of the operand
ALGN(n) means padding with null bytes to an n-byte boundary

initramfs  := ("\0" | cpio_archive | cpio_gzip_archive)*

cpio_gzip_archive := GZIP(cpio_archive)

cpio_archive := cpio_file* + (<nothing> | cpio_trailer)

cpio_file := ALGN(4) + cpio_header + filename + "\0" + ALGN(4) + data

cpio_trailer := ALGN(4) + cpio_header + "TRAILER!!!\0" + ALGN(4)
```

In human terms, the initramfs buffer contains a collection of compressed and/or uncompressed cpio archives (in the "newc" or "crc" formats); arbitrary amounts zero bytes (for padding) can be added between members.

The cpio "TRAILER!!!" entry (cpio end-of-archive) is optional, but is not ignored; see "handling of hard links" below.

The structure of the cpio_header is as follows (all fields contain hexadecimal ASCII numbers fully padded with '0' on the left to the full width of the field, for example, the integer 4780 is represented by the ASCII string "000012ac"):

Field name	Field size	Meaning
c_magic	6 bytes	The string "070701" or "070702"
c_ino	8 bytes	File inode number
c_mode	8 bytes	File mode and permissions
c_uid	8 bytes	File uid
c_gid	8 bytes	File gid
c_nlink	8 bytes	Number of links
c_mtime	8 bytes	Modification time
c_filesize	8 bytes	Size of data field
c_maj	8 bytes	Major part of file device number
c_min	8 bytes	Minor part of file device number
c_rmaj	8 bytes	Major part of device node reference
c_rmin	8 bytes	Minor part of device node reference
c_namesize	8 bytes	Length of filename, including final 0
c_chksum	8 bytes	Checksum of data field if c_magic is 070702; otherwise zero

The c_mode field matches the contents of st_mode returned by stat(2) on Linux, and encodes the file type and file permissions.

The c_filesize should be zero for any file which is not a regular file or symlink.

The c_chksum field contains a simple 32-bit unsigned sum of all the bytes in the data field. cpio(1) refers to this as "crc", which is clearly incorrect (a cyclic redundancy check is a different and significantly stronger integrity check), however, this is the algorithm used.

If the filename is "TRAILER!!!" this is actually an end-of-archive marker; the c_filesize for an end-of-archive marker must be zero.

Handling of hard links

When a nondirectory with c_nlink > 1 is seen, the (c_maj,c_min,c_ino) tuple is looked up in a tuple buffer. If not found, it is entered in the tuple buffer and the entry is created as usual; if found, a hard link rather than a second copy of the file is created. It is not necessary (but permitted) to include a second copy of the file contents; if the file contents is not included, the c_filesize field should be

set to zero to indicate no data section follows. If data is present, the previous instance of the file is overwritten; this allows the data-carrying instance of a file to occur anywhere in the sequence (GNU cpio is reported to attach the data to the last instance of a file only.)

c_filesize must not be zero for a symlink.

When a "TRAILER!!!" end-of-archive marker is seen, the tuple buffer is reset. This permits archives which are generated independently to be concatenated.

To combine file data from different sources (without having to regenerate the (c_maj,c_min,c_ino) fields), therefore, either one of the following techniques can be used:

a. Separate the different file data sources with a "TRAILER!!!" end-of-archive marker, or
b. Make sure c_nlink == 1 for all nondirectory entries.