NanoFS v1.0

File System Specification rev. 1 (Draft)

Paulino Ruiz de Clavijo Vázquez <paulino@dte.us.es>
Enrique Ostúa Aragüena <ostua@dte.us.es>

1. About NanoFS

The NanoFS (Nano Filesystem) has been developed as a very simple file system aimed to be implemented as IPCore in custom hardware designs. It has been designed from scratch keeping in mind the hardware implementation limitations.

To reduce hardware resources a new internal layout of file system is proposed to optimize file system data structures. The main feature is that the required data to navigate across file system layout and the file contents is mixed. Each single data block in the storage device contains both parts. With this characteristic when part of file data is fetched, the retrieved data contain a piece of file content and the required information about of location of next part. This distinguishes it from the other file systems.

The new file system layout divides the storage device in data blocks structured as nodes of a linked list. The file system is optimal when node size and block size in the storage device matches. All nodes of the file system are in a forward linked list, being all data on device reachable. Internally all nodes contains two parts: fields with pointers to other nodes and a field with the file data. The file data field has a fragment of the file contents. To simplify the structures management, only exists two types of data node. One is called data node and is used to store file content. The other node type represents directory entries and is used for directories tree structure.

2. File system layout

NanoFS consists of a data structures stored along the memory device. As in most filesystems, the block device is divided in blocks of fixed size called *block size filesystem*. With NanoFS each block is refered by a unique integer number called BlockNo. The block numbers are sequential starting in 0.

A NanoFS formated device is built as a linked list of nodes of NanoFS data structures. Each node of the filesystem is stored in one or several straight blocks. The filesystem is stuctured in a linked list where nodes are referenced using its blockNo.

Besides superblock with NanoFS a node contains a dir_node or data_node. The figure 1 depicts a nodes forming a filesystem with nodes offsets (in bytes) and pointers referencing blockNos.

Main remarks about filesystem nodes and structs:

- Data is stored in little endian format

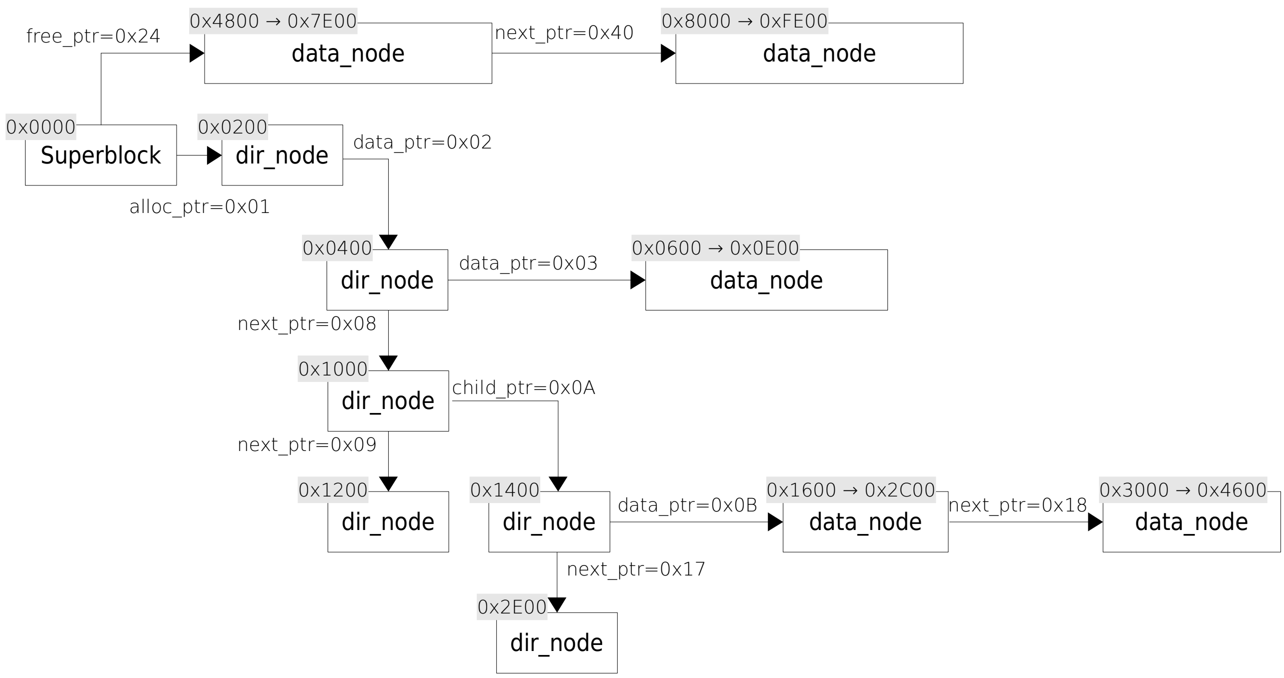

Figure 1. Example of NanoFS layout using 512bytes of block size.

2.1. Superblock

The starting point of NanoFS is the superblock located at byte offset 0 of memory/device. Superblock links with two data structures: data blocks allocated and free data blocks. Field block size represents the field length multiplier

Offset in bytes	Size in bytes	Name	Descrition
0	2	s_magic	Magic Number 0x4E61
2	1	s_blocksize	Block size
3	1	s_revision	Revision
4	4	s_alloc_ptr	Absolute blockNo of allocated root entry
8	4	s_free_ptr	Absolute blockNo of start of free blocks list
12	4	s_fs_size	Filesystem size in blocks
16	16	s_uuid	Fixed NanoFS UUID
32	2	s_extra_size	Extra superblock size in bytes

Table 1. Superblock data structure (nanofs_superblock).

Block size value	Block size	Max file system size
0	1 byte	4GiBytes
1	512 bytes	2TiBytes
2	4096 bytes	16TiBytes

Table 2. Block size field.

2.2. Allocated block data structure

Allocated blocks data structure is used to store directory structure and all content of files. This information is kept using three types of nodes:

- Directory entry nodes
- Metadata nodes
- Data nodes

As seen above (fig 1), allocated blocks begin at root node linked from superblock. This root node is of type directory entry and data contains filesystem label.

2.3. Directory entry structure

In the directory entries, pointers are 4 bytes length and its represent a absolute block number starting in 0. The block number 0 is superblock. All pointers/block numbers are in little endian format.

Offset in bytes	Size in bytes	Field name	Description
0	1	d_flags	Directory entry flags
1	4	d_next_ptr	Absolute blockNo of next directory entry
5	4	d_data_ptr	Absolute blockNo of first child data block
9	4	d_meta_ptr	Absolute blockNo of first metadata block
13	1	d_fname_len	Length in bytes of filename
14	256	f_name	Name of file
270	34	f_meta	Standard metadata (see table 5)

Table 3. Directory entry structure (dir_node).

Directory entry field d_flags details:

	7	6	5	4	3	2	1	0
d_flags	f_metadata	-	-	-	-	f_type2	f_type1	f_type0
	1: Valid standard metadata 0: No metadata					000: Directory 001: Reg. File 010: Character device 011: Block device 100: Fifo 101: Socket 110: Sym. Link		

Table 4. Bits for d_flags byte.

As far as possible, the *metadata* structure uses the same fields and size that the ext2/ext4 () filesystem inodes structures.

Standard metadata			
Offset in bytes	Size in bytes	Field name	Description
270	4	m_uid	32 bits owner user ID
274	4	m_gid	32 bits group ID
278	4	m_atime	32bit, the last time this file was accesed (number of seconds since january 1st 1970)

Standard metadata			
282	4	m_ctime	32bit, time when the file was created (number of seconds since january 1st 1970)
286	4	m_mtime	32bit, the last time when this file was modified (number of seconds since january 1st 1970)
290	4	m_atime_extra	See ref. [2]
294	4	m_ctime_extra	See ref. [2]
298	4	m_mtime_extra	See ref. [2]
302	2	m_mode	Based in ext2/ext4 i_mode field with some limitations

Table 5. Standard metadata.

7	6	5	4	3	2	1	0
S_IWUSR	S_IXUSR	S_IRGRP	S_IWGRP	S_IXGRP	S_IROTH	S_IWOTH	S_IXOTH
Owner may write	Owner may execute	Group members may read	Group members may write	Group members may execute	Others may read	Others may write	Others may execute

Table 6. Bits for mode, lower byte (field m_mode, offset 302)

15	14	13	12	11	10	9	8
–	–	–	–	S_ISUID	S_ISUID	S_ISVTX	S_IRUSR
				Set UID	Set GID	Sticky bit	Owner may read

Table 7. Bits for mode node, upper byte (field m_mode, offset 303).

2.4. Data block and free blocks data structure

The following structure is used for data stored nodes and free nodes.

Offset in bytes	Field name	Size in bytes	Field name
0	d_next_ptr	4	Absolute next block pointer
4	d_len	4	Data length
8	d_data	d_len	Data

Table 8. Data nodes structure.

3. File system limits

- Max file name: 255 characters
- Hard links not supported

4. References

[1] The Second Extended File System, Internal Layout, Dave Poirier, <instinc@gmail.com>
[2] Ext4 Disk Layout, <https://ext4.wiki.kernel.org/index.php/Ext4_Disk_Layout>