Haribote File System Specification

Version 1.0

Written by Allen He

Notices:
This document may contain errors, and the author bears no full responsibility for these errors.

The author reserves the right of final interpretation of this document.

Thanks Hidemi KAWAI, for guiding so many beginners in writing operating system.

Contents

1

Introduction

Haribote File System (HAFS) is a lightweight file system designed for the Haribote OS and Haribote OS for Metaverse.

HAFS inherits the minimalist style of Haribote OS, but mature file systems contain complex structures. Therefore, to enhance robustness and utility, HAFS must incorporate certain moderately complex structures.

HAFS consists of a Boot Sector, a Super Sector, bitmaps and segments. The smallest unit of storage allocation is the block. And each segment contains a fixed number of contiguous blocks. Every segment consists of Segment Super Sector, segment block maps, segment inode maps, inodes and data blocks.

Like Unix-like operating systems' file system, HAFS uses inodes to record file information. But HAFS's inodes are smaller, and other attributes are recorded in additional blocks.

This document provides a detailed technical specification of HAFS, covering its structure, implementation, and parameters.

2

Structure

2.1 Overall Structure

Boot Sector	Reserved sectors	Super block	Block map	Inode map	Segment 0	...

Table 1: The structure of HAFS.

Table 1 shows the structure of HAFS. The partition is divided into segments. There are five special parts in the partition: Boot Sector, Reserved sectors, Super block, Global block bitmap and Global inode map. Despite being global in scope, these structures are physically stored within segment 0.

The global block bitmap identifies segments with free blocks; likewise, the global inode bitmap identifies segments with free inodes.

Segment Super Sector	Block map	Inode map	Inodes	Data blocks

Table 2: The structure of a segment.

Each segment begins with Segment Super Sector, next are block map, inode map and inodes. The remaining part is the data blocks.

Unlike other segments, the Segment Super Sector of segment 0 resides following the global inode bitmap.

2.2 Boot Sector

Following is the structure of Super Sector:

Offset	**Name**	**Length**	**Describe**
0x0	Jump instruction	3	Jump instruction
0x3	OEM code	8	Set to ”HAFS ”
0xB	Sector size	2	Sector size in bytes
0xD	Block size	1	Block size in sectors
0xE	Reserved sector	2	Reserved sector number
0x10	Unused	4	Unused
0x14	Media descriptor byte	2	Disk type, 0xF8 for Harddisk
0x16	Unused	4	Unused
0x1A	Signature	4	The signature, set to 0x00800080
0x1E	Sector number	8	Sector number in total
0x26	Version	4	Version number, set to 0x1
0x2A	Super Sector start	4	The start sector* of the Super Sector
0x2E	Boot code	—	Boot code
0x1FE	Boot sector signature	2	0xAA55

*: The starting sector is relative to the partition.

2.3 Super Sector

Following is the structure of Super Sector:

Offset	Name	Length	Describe
0x0	Signature	4	”HAFS”
0x4	Partition size	8	Partition size in byte
0xC	Block size	2	Block size in byte
0xE	Block map start	2	Block map start sector*
0x10	Inode map start	2	Inode map start sector*
0x14	Segment 0 start	4	Segment 0 start sector*
0x18	Root inode	4	Root directory’s inode
0x1C	Disk type	4	Disk type
———	0x1		Hard disk
———	0x2		CD / DVD
———	0x4		Floppy
———	0x8		USB Flash Disk
———	0x10		Bootable
———	0x20		Hidden
———	0x40		Read-only
0x20	Inode per segment	4	Inode number per segment
0x24	Segment number	4	Segment number in total
0x28	Block map size	4	Block map size in byte
0x2C	Inode map size	4	Inode map size in byte
0x30	Check code	4	CRC32 check code

*: The starting sector is relative to the partition.

2.4 Segment Super Sector

Following is the structure of Segment Super Sector:

Offset	**Name**	**Length**	**Describe**
0x0	Signature	4	”HAFS”
0x4	Block number	2	The block number of this segment
0x6	Block map start	2	Block map start sector*
0x8	Inode map start	2	Inode map start sector*
0xA	Inode area start	2	Inode area start sector*
0xC	Free block number	4	Free block number
0x10	Free inode number	4	Free inode number
0x14	Check code	4	CRC32 check code

*: The starting sector is relative to the **segment**.

2.5 Inode

Following is the structure of inode:

Offset	**Name**	**Length**	**Describe**
0x0	Type	4	The type of this inode
———	0x1		Present
———	0x2		File
———	0x4		Directory
———	0x8		Hidden
———	0x10		Read-only
———	0x20		System
———	0x40		With long filename
———	0x80		With additional attribute
0x4	Size	8	File size in byte
0xC	Additional attribute block	4	*
0x10	Direct blocks	12*4	*
0x40	L1 indirect blocks	8*4	*
0x60	L2 indirect blocks	4*4	*
0x70	L3 indirect blocks	4*4	*

*: The starting block is relative to the partition.

2.6 Directory

Following is the structure of directory:

Offset	**Name**	**Length**	**Describe**
0x0	Next record	4	The offset of next record*
0x4	Inode	4	The inode of the file
0x8	Name	—	File name

*: The offset is relative to this record.

2.7 Additional Block

Following is the types of additional attribute:

Number	**Name**	**Describe**
0x2	Long name	The long name of file
0x8	Father inode	The inode of parent directory
0x10	Create time	Create time
0x20	Access time	Access time, only for files
0x40	Modify time	Modify time, only for files

3

Operating Specification

3.1 Format

To format a partition to HAFS, follow the steps below:

1. Collect and calculate the necessary parameters, including segment number, the starting sector numbers of maps and the starting sector number of the zero segment.

(After completing each of the following steps, write the changes into the partition immediately.)

2. Construct the Boot Sector.

3. Construct the Super Sector, fill the area preceding the check code, and then calculate the check code for the filled content.

4. Set the global block map and the global inode map to zero.

5. Construct the Segment Super Sector of the zero segment, fill the area preceding the check code, and then calculate the check code for the filled content.

6. Set the block map and the inode map for the zero segment; the bits corresponding to the blocks used (i.e. the blocks before the zero segment, the Segment Super Sector and maps) should be set.

7. Construct the root inode. Set the inode type ”Present” and ”Directory”.

8. Construct the Segment Super Sectors and maps of all other segments.

3.2 Create a file

To create a file, follow the steps below:

1. Check the validity of the destination path.

2. Get the inode number of the destination directory.

3. Traverse the directory entries to check for any duplicate files.

3. Allocate a new inode and add a new record to the destination directory.

4. Initialize the inode and fill the create time, the access time and the modify time.

3.3 Create a directory

To create a directory, follow the steps of creating a file. Notice that directories don't have the access time and the modify time.

3.4 Delete a file

To delete a file, follow the steps below:

1. Check if the file exists.

2. Get the inode number of the file and the file's parent directory.

3. Remove the record of the file in the parent directory.

4. Free all blocks allocated by the file.

5. Set the inode of the file to zero.

6. Free the inode of the file.

3.5 Delete a directory

To delete a directory, follow the steps of deleting a file. Notice that directories should be empty.

3.6 Read a file

To read a file, check if the file exists first, then read the data of the file, and change the access time of the file to current time at last.

3.7 Write to a file

To write a file, follow the steps below:

1. Check if the file exists.

2. Calculate the number of the free blocks, and then check if it is enough to allocate blocks for the new data.

3. Change the access time and the modify time of the file to current time.

4. Allocate new blocks for the file.

5. Put the data to the blocks.

3.8 Notes

In all operations except the format, write the changes to the buffer of the operating system if it exists, or write them into disk immediately.

The number of free blocks/inodes should be checked before allocation.

The access and modify time should be changed in advance to prevent accidental loss.

4

Parameter Specification

Here are some fixed parameters of HAFS:

1. Block size: 1024/2048/4096 bytes
2. Segment size: 1024 blocks
3. Maximum partition size: LLONG_MAX KiB

Here are some alignment requirements:

The Super Sector and the Segment Super Sectors: Align to sector.

Maps: Align to sector.

Zero segment start: Align to block.

Data part: Align to block.