Siaranite Disk File System

A block file system written and implemented in C#

primarily for Cosmos Operating Systems

SDFS Version 0.0.1 Specification Version 1.0.0

Table of contents

* [How to use this document](#_How_to_use)
* [About SDFS](#_About_SDFS)
* [Existing projects using SDFS / SDFS-Compliant Operating System projects](#_Existing_projects_using)
* [Objects](#_Objects)
* [Methods](#_Methods)
* [Properties](#_Properties)
* [Definitions](#_Definitions)
  + [Block](#_Block)
  + [Entry Attributes](#_Entry_Attributes)
  + [Entry](#_Entry)
  + [File Entry](#_File_Entry)
  + [Directory Entry](#_Directory_Entry)
  + [File System](#_File_System)
* [How to use the source code](#_How_to_use_1)
* [Source Directory Structure](#_Source_Directory_Structure)
* About this document
* [Licensing Information](#_Licencing_Information)

How to use this document

This document will be the main piece of documentation to be used when implementing the Siaranite Disk FileSystem in your operating system or software project. There will be some places where terms will be abbreviated, and their definitions/meaning will be stated in this section.

* SDFS - Siaranite Disk File System
  + The Siaranite Disk File System is this FileSystem, its implementation and according documentation. SDFS is being maintained cooperatively by Siaranite Solutions, Aura Systems, and its contributors.
* Cosmos – C# Open Source Managed Operating System
  + The operating system development project upon which SDFS is built.
* Medli OS
  + The reference operating system implementation which SDFS is built for. It was created by Arawn Davies and is being maintained cooperatively by Siaranite Solutions and its contributors.
* Aura OS
  + Aura is an operating system written in C#, created by Valentin Charbonnier and Alexy Da Cruz. Aura I developed by the Aura Systems team and its contributors. SDFS has also been developed with and for Aura to work closely with Medli and other operating systems.

About SDFS

SDFS is a filesystem and logical drive address system written in C# that was made for the Medli and Aura Operating Systems. It uses logical objects called “blocks”, which extend upon the blocks used in a variety of hard disk drives and other non-volatile storage media. The original implementation is written in C# but future implementations and drivers are planned to be written in other languages such as C and C++. This will allow SDFS volumes to be written using many more operating systems on a variety of different platforms.

Existing projects using SDFS / SDFS-Compliant projects

The first projects to use SDFS include Medli OS, an operating system written by Siaranite Solutions, the developer of SDFS, and Aura OS, an operating system developed and maintained by Aura Systems. An implementation of the FAT filesystem was already built into these operating systems as part of the Cosmos project, but support for a custom filesystem built specifically for these two operating systems was planned.

Objects

Block

The filesystem is logically divided into blocks based on the block count and block size of the storage media it is applied onto. These blocks are then used to hold files and directories onto.

Entry

Each entry on the filesystem, where it is a directory containing other directories and files, or files, is comprised of one or more blocks. Each entry will have a path and name, with the full name returning the path followed by the name at the end.

File Entry

When the user wishes to store file contents on a disk, whether it is an array of bytes representing executables or strings representing text, a file entry is used to represent the blocks holding the data as one single entity.

Directory Entry

A directory entry serves a similar purpose as a file, but instead of representing string or byte data, it is a structure containing other directories and/or files. Directories are used to store data in an organised way, making the retrieval of other directories and files more quick and accessible instead of having loose disorganised data stored on the root level of a filesystem.

Filesystem

The filesystem is a logical representation of the data stored on the storage media. A 64-bit byte sequence is used for a filesystem signature in addition to the system ID stored in the Master Boot Record of a hard disk drive on x86 platforms. This byte array consists of the following bytes:

0x4D, 0x65, 0x64, 0x6C, 00x69, 0x44, 0x46, 0x53.

This byte when represented using ASCII encoding after converting from byte to char, returns the string “MedliDFS”. This will help identifying an SDFS partition when other partitions reporting the same partition ID exist. Byte signature verification should only be used when there is no other way to determine whether a partition uses a SDFS filesystem.

Methods

Here are the methods and the specified arguments currently used in the logical implementation of SDFS. These are subject to change in future revisions of any specification, documentation and implementation of SDFS.

|  |  |  |
| --- | --- | --- |
| class Block | | |
| Method name | Method parameters | Method return type |
| Block | Array of Bytes, Partition, ulong | Constructor |
| Read | Partition, ulong | Block |
| Write | Partition, Block | void |
| GetFreeBlock | Partition | Block |

|  |  |  |
| --- | --- | --- |
| class FileEntry | | |
| Method name | Method parameters | Method return type |
| File | - | Constructor |
| WriteAllBytes | Array of Bytes | void |
| WriteAllText | String | void |
| ReadAllBytes | - | Array of Bytes |
| ReadAllText | - | String |

|  |  |  |
| --- | --- | --- |
| class DirectoryEntry | | |
| Method name | Method parameters | Method return type |
| DirectoryEntry | Partition, ulong, String | Constructor |
| RetrieveDirectories | - | Array of Directories |
| RetrieveFiles | - | Array of Files |
| RetrieveEntries | - | Array of Entries |
| AddDirectory | String | void |
| GetDirectory | String | Directory |
| GetFile | String | File |

|  |  |  |
| --- | --- | --- |
| class Entry | | |
| Method name | Method parameters | Method return type |
| EditAttributes | EntryAttribute, long | void |
| CreateEntry | Partition, Block, String | Block |
| CreateEntry | Partition, String | Block |

|  |  |  |
| --- | --- | --- |
| class FileSystem | | |
| Method name | Method parameters | Method return type |
| FileSystem | Partition | Constructor |
| MapFS | FileSystem | void |
| JoinFilename | String, String | String |
| JoinDirectoryName | String, String | String |
| GenerateFS | Partition | Boolean value |
| GenerateFS | - | Boolean value |
| ValidateFS | - | Boolean value |
| Refresh | Ulong | void |
| Clean | Block | void |

Properties

|  |  |  |
| --- | --- | --- |
| class FileSystem | | |
| Property Name | Return type | Keywords w/ value |
| \_partition | Partition | Private, Partition |
| Separator | String | Private |
| fsSignature | Byte array | Private |
| BlockSize | Unsigned long | \_partition.BlockSize |
| BlockCount | Unsigned long | \_partition.BlockCount |
| RootDirectory | Directory | Public, new Directory(\_partition, 1, separator) |

|  |  |  |
| --- | --- | --- |
| class Entry | | |
| Property Name | Return type | Keywords w/ value |
| UnacceptableChars | Char array | Protected static |
| startBlock | Block | Protected |
| \_path | String | Private |
| mPath | String | Public, \_path |
| \_partition | Partition | Private |
| mPartition | Partition | Public, \_partition |
| MaxFilenameLength | Integer | Private static, 255 |
| Name | String | Public |
| CreateEntry(Partition, String) | Block | Protected static |
| CreateEntry(Partition, Block, String) | Block | Protected static |

|  |  |  |
| --- | --- | --- |
| class Block | | |
| Property Name | Return type | Keywords w/ value |
| MaxContentSize | Unsigned Integer | Private |
| \_partition | Partition | Private |
| Partition | Partition | Public, \_partition |
| \_blockNumber | Unsigned long | Private, Public BlockNumber |
| \_blockCount | Unsigned long | Private, Public BlockCount |
| \_cSize | Unsigned long | Private, Public ContentSize |
| \_tSize | Unsigned long | Private, Public TotalSize |
| \_nBlock | Block | Private, Public NextBlock |
| \_Used | Boolean | Private, Public Used |
| Content | Byte array |  |

|  |  |  |
| --- | --- | --- |
| class Directory | | |
| Property name | Return type | Keywords w/ value |
| DirectoryName | String |  |
| ToString | String |  |

|  |  |  |
| --- | --- | --- |
| class File | | |
| Property name | Return type | Keywords w/ value |
| Filename | String | Public |

How to use the source code

Before deciding to implement SDFS in your operating system or other hardware/software project, it is important to understand what this code uses, how it runs and how portable/cross-platform it is before adding it to a different software project than what is has been developed for. By the time this this specification document is published, only an implementation written in C# is available until a driver is written using different program languages and compilers such as C and C++. Other compatible drivers are planned for the future, but work will be done in C# to allow use in Medli and Aura operating systems. If your project is still in early stages of development, then you will want to review the current development status of this project including its stability, in case of any bugs or unexpected events occurring once deployed.

Source Directory Structure

* /docs
  + Specification.rtf - This Document
  + Licence.txt - SDFS Licence Information
* /source
  + /SDFS - Driver C# Implementation
    - /Logical - Logical FS definitions
      * /Entries - Filesystem Entry definitions
        + Directory.cs - Directory class
        + File.cs - File class
        + Entry.cs - Entry class
      * Block.cs - Block class
      * Filesystem.cs - Filesystem class
    - /Physical - Physical device & class definitions
      * IDE.cs - IDE class
      * MBR.cs - Master Boot Record class
      * Partitions.cs - Partition class
      * PrimaryPartition.cs - Primary Partition class
    - SDFS.csproj - C# Project File
    - UtilityMethods.cs - Utility Method definitions
  + /SDFS-Demo - Driver Demonstration in C#
  + SDFS.sln - Visual Studio 2018 Solution for source code

Licensing Information

SDFS is released under the following BSD 3-Clause license:

BSD 3-Clause License

Copyright (c) 2018, Siaranite Solutions

All rights reserved.

Redistribution and use in source and binary forms, with or without

modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice,

this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,

this list of conditions and the following disclaimer in the documentation

and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its contributors

may be used to endorse or promote products derived from this software

without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"

AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE

IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE

FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL

DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR

SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER

CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,

OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE

OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.