SFS (Simple File System) Specification

Draft Version 1.0

December 7, 2006

NOTICE

Table of Contents:

Introduction

Unlike networking protocols, there is currently no widely accepted format for freely exchanging data on physical media between computers and other devices. Instead people have been using a variety of file systems that are either not widely accepted or subject to technical, licensing, copyright or patent restrictions.

The primary purpose of the Simple File System is to facilitate the free exchange of data on physical media without unnecessary restrictions or complexity. For this reason the Simple File System has been designed in such a way that it is easy to understand and implement.

File System Format Overview

The file system uses different areas on the physical media for different purposes. The five areas are:

- the super-block
- the reserved area
- the data area
- free area
- the index area

The purpose and usage of each of these areas is described in detail in the following sections. These areas are always in the following order:

Super-block
Reserved area
Data Area
Free Area
Index Area

Table 1 - Layout of areas on the media

All values defined within this specification must be stored with the least significant byte at the lowest address (i.e. in "little-endian" format). All fields marked as "Unused/reserved" must be filled with zeros when written to the media and ignored when read from the media, except as otherwise noted (in the super-block and reserved area).

Super-Block Format

The super-block contains values needed to determine the physical size and layout of the media itself, and values needed to determine the location of each area. The super-block also contains many values that are used for compatibility purposes.

Offset	**Size in bytes**	**Description**
0x0000	11	Reserved for compatibility (boot code)
0x000B	21	Reserved for compatibility (legacy BIOS parameter block)
0x0020	372	Reserved for compatibility (boot code)
0x0194	8	Time stamp
0x019C	8	Size of data area in blocks
0x01A4	8	Size of index area in bytes
0x01AC	3	Magic number (0x534653)
0x01AF	1	Simple File System version number (0x10 for Version 1.0)
0x01B0	8	Total number of blocks
0x01B8	4	Number of reserved blocks
0x01BC	1	Block size
0x01BD	1	Checksum
0x01BE	64	Reserved for compatibility (partition table)
0x01FE	2	Reserved for compatibility (boot signature)

Table 2 - Super-block format

The reserved areas at offsets 0x0000, 0x000B, 0x0020, 0x01BE and 0x1FE are used for compatability purposes, and is not covered by this specification. Data in these areas must be ignored when the super-block is read, and never modified (unless the modifications are for purposes not covered by this specification).

The checksum field (at offset 0x01BD) is calculated such that the sum of all bytes stored between offset 0x01AC and 0x01BD inclusive will have the lowest 8 bits clear. This checksum is set when the media is formatted and does not normally need to be changed after the media is formatted (however, it may need to be recalculated if the version number at offset 0x01AF is changed).

The block size (at offset 0x01BC) is used to calculate the number of bytes in each block. The formula for this calculation is "bytes_per_block = $2^{(block\_size + 7)}$", so that a value of 2 represents 512 bytes per block, 3 represents 1024 bytes per block, etc. This value must be equal to or greater than 1 (128 bytes per block or less is not supported). For media that is organized into "sectors" (floppy disks, hard disks, etc) the block size must match the size of a sector. For example, for a common 1440 KB floppy disk this value would be 2 (512 bytes per sector).

The number of reserved blocks (at offset 0x01B8) specifies the number of blocks that are reserved for the reserved area and the super-block. This value is also used

to determine where the data area begins. This value must be 1 or greater (as it includes the super-block). For example, if the first block is "block number 0" and the reserved area (including the super-block) is 20 blocks in size, then this value must contain 20 and the first block in the data area will be "block number 20".

The total number of blocks (at offset 0x01B0) includes all areas defined in this specification, including the super-block and reserved area. It is used to determine the physical size of the media and for finding the end of the index area. The formula for calculating the physical size of the media in bytes is "media_size = total_blocks * $2^{(block\_size + 7)}$". For example, for a common 1440 KB floppy disk this value would be 2880.

The magic number (at offset 0x01AC) is used to indicate compliance with this specification. The magic number 0x534653 represents the ASCII characters 'SFS'. The magic number is positioned next to the version number so that software can check both in a single operation (for example, software can check if the 32 bit value at offset 0x01AC equals 0x10534653).

The Simple File System version number (at offset 0x01AF) must contain the value 0x10 to indicate compliance with this specification. This value is in fixed point BCD, where the high 4 bits determine the major version number and the lower 4 bits determine the minor version number. Any future versions of this specification (which aren't expected but are possible) will use numerically higher values for the version number. For example, 0x23 would represent version 2.3 of the SFS specification.

All future versions of the Simple File System will remain compatible with "SFS Version 1.0", in that optional features may be added. If the version number (at offset 0x01AF) is numerically higher than the value 0x10, then code that implements "SFS Version 1.0" can leave the version number unchanged and provide read-only access to file data. This preserves any optional features added in later versions of SFS. Alternatively, code that implements "SFS Version 1.0" can replace the version number with 0x10 to automatically revert the file system such that it complies with this specification (any data in the file system that belongs to optional features added in later versions of SFS become discarded/ignored). In this case write access can be permitted (however it is recommended that the version number remain unchanged until the first modification is made, rather than changing it when the file system is "mounted"). Please note that in order to change the version number, the checksum field (at offset 0x01BD) must also be recalculated.

The size of the index area (at offset 0x01A4) indicates the size of the index area in bytes, and is also used to calculate the location of the start of the index area by subtracting the size of the index area from the total size of the media. This value is also used to determine the end of the free space area by calculating the start of the index area and rounding it down to the nearest block size. The size of the index area must be a multiple of 64 bytes.

The size of the data area (at offset 0x019C) indicates how large the data area currently is in blocks. It does not indicate how many blocks within the data area are actually used (for example, if files have been deleted it is likely that some blocks within the data area are not in use).

The time stamp field (at offset 0x0194) indicates when the size of the index area (at offset 0x01A4) or the size of the data area (at offset 0x019C) was changed last. This field is used is used in conjunction with the Volume Identifier to improve the detection of "media changed" on some hardware (e.g. old floppy drives). The time stamp is stored in the same format as all time stamps used by the Simple File System.

Reserved Area

The reserved area (immediately following the super-block) may be used for any number of things that are outside the scope of this specification. Examples would include additional file information, boot code or data used for compatibility. All data within the reserved area must be ignored by all software that complies with this specification (i.e. even though it's reserved, it's never cleared when written to because it's never written to). The size of the reserved area can be determined from "number of reserved blocks" field (at offset 0x01B8) in the super-block.

It should be noted that the "number of reserved blocks" field (at offset 0x01B8 in the super-block) includes the super-block itself. This means that for 256 byte blocks this value must be 2 or higher, and for larger block sizes this value must be 1 or higher. The largest possible value in the "number of reserved blocks" field (at offset 0x01B8) can be calculated by subtracting 2 blocks (one for the index area and another for the data area) from the "Total number of blocks" field (at offset 0x01B0 in the super-block), unless the result is too large for a 32 bit unsigned integer (in which case the reserved area is limited to 0xFFFFFFFF blocks).

Data Area Format

The data area is used to store all file data. The start of the data area is determined by the "number of reserved blocks" field in the super-block, and the end of the data area is determined by adding the "size of data area" and "number of reserved blocks" fields.

The data for any specific file consumes sequential blocks in the data area (file fragmentation is not supported). The data area may contain unused blocks, where files have been deleted or reduced in size, or unused blocks have been deliberately left to allow data to be appended to the preceding file. Different techniques for managing the blocks within the data area are discussed in Appendix A.

Free Area

The free area is an area of unused blocks between the data area and index area. As new files are added to the file system, the data area below the free area grows and the index area above the free area grows, causing the free area to shrink from both

ends.

The start of the free space area is determined by adding the "size of data area" and "number of reserved blocks" fields. The end of the free space area is determined by subtracting the size of the index area from the total size of the media and rounding it down to the nearest block size. It is assumed that these values would be calculated during initialization and updated in software after that.

Index Area Format

The index area is used to manage the usage of blocks in the data area and keep track of file names. The start of the index area is determined by subtracting the size of the index area from the total size of the media. The end of the index area is determined by the total size of the media.

The index area contains a variable number of 64 byte entries, where every file has one or more entries (but every entry doesn't necessarily represent a file). The format for these entries is described in the following section. The total number of entries stored in the index area can be determined by dividing the size of the index area by 64. Entries are not sorted in any way, except for continuation entries (which always immediately follow the entry they continue). For directories, entries for '.' and '..' are never stored in the file system and are assumed to exist when necessary.

There are 9 different types of index area entries:

- Volume Identifier (type 0x01)
- Starting Marker Entry (type 0x02)
- Unused Entry (type 0x10)
- Directory Entry (type 0x11)
- File Entry (type 0x12)
- Unusable Entry (type 0x18)
- Deleted Directory Entry (type 0x19)
- Deleted File Entry (type 0x1A)
- Continuation Entry (any value between 0x20 and 0xFF inclusive)

These entry types are described in the following sections.

Any index entry type not listed above must be treated as an unused entry (0x10), and may be replaced with the entry type value 0x10 by an SFS 1.0 compliant code at any time.

Some of these entry types have specific positions within the index area. For example:

Last block in free area
Empty space (present whenever the start of the index area isn't aligned to the

Table 3 - One possible index area

start of a block)
Starting marker entry
File entry 5
Unused entry
File entry 4
File entry 3
Continuation entry for file entry 4
Deleted file entry
Continuation entry for deleted file entry
Unusable entry
File entry 2
Directory entry 1
Continuation entry for directory entry 1
File entry 1
Volume identifier entry
End of media

Index Area - Volume Identifier

The very first entry (just below the end of the media) in the index area must be a Volume Identifier Entry, which is used to detect if the media is changed on devices that don't have a reliable "media changed" facility (e.g. old floppy drives). There must only be one Volume Identifier Entry at the end of the media.

The Volume Identifier Entry uses the following format:

Offset	**Size in bytes**	**Description**
0x00	1	Entry type (0x01 for the Volume Identifier Entry)
0x01	3	Unused/reserved (must be zero)
0x04	8	Time stamp
0x0C	52	Volume name in UTF-8, including zero terminator

Table 4 - Volume Identifier Entry format

The entry type is used to determine that this is a Volume Identifier Entry and not any other type of index area entry.

The time stamp field indicates when the media was first formatted (it is not changed during normal operation). It is stored in the same format as all time stamps used by the Simple File System.

The volume name is limited to 52 bytes (including the terminator), and some characters are not allowed within a volume name. This name field follows the format for all name fields in the Simple File System.

Index Area - Starting Marker Entry

The Starting Marker Entry is used to find the beginning of the index area during file system recovery (i.e. when the super-block is unusable). There must always be one Starting Marker Entry, and there must never be any index area entries closer to the start of the media than the Starting Marker Entry.

The Starting Marker Entry uses the following format:

Offset	Size in bytes	Description
0x00	1	Entry type (0x02 for the Starting Marker Entry)
0x01	63	Unused/reserved

Table 5 - Starting Marker Entry format

The entry type is used to determine that this is a Starting Marker Entry and not any other type of index area entry.

Index Area - Unused Entry

Unused Entries are used to mark an index area entry as unused (not to mark blocks within the data area as unused).

The format for an Unused Entry is:

Offset	Size in bytes	Description
0x00	1	Entry type (0x10 for Unused Entries)
0x01	63	Unused/reserved

Table 6 - Unused Entry format

The entry type is used to determine that this is an Unused Entry and not any other type of index area entry.

Index Area - Directory Entry

Each Directory Entry in the index area uses the following format:

Offset	Size in	Description

Table 7 - Directory Entry format

	bytes	
0x00	1	Entry type (0x11 for Directory Entries)
0x01	1	Number of continuation entries following this entry
0x02	8	Time stamp
0x0A	54	Directory name in UTF-8

The entry type is used to determine that this is a Directory Entry and not any other type of index area entry.

The number of continuation entries is used to determine how many Continuation Entries immediately follow an entry when the directory name is too long to store in the Directory Entry alone. A value of zero indicates that there are no Continuation Entries. See the description of Continuation Entries for more information.

The time stamp field indicates when the Directory Entry information was last modified. It is stored in the same format as all time stamps used by the Simple File System. Optionally, the time stamp within a Directory Entry may be updated when a file or sub-directory within the associated directory is created, deleted, moved or renamed.

The directory name field is used to store the name of the directory, including the path. For example, for the directory "foo/bar" this field will contain the characters "foo/bar" followed by a zero to mark the end of the string (not just "bar"). If the directory name is too long to store in the Directory Entry, Continuation Entries are used. See the description of Continuation Entries for more information.

Index Area - File Entry

Each File Entry in the index area uses the following format:

Offset	**Size in bytes**	**Description**
0x00	1	Entry type (0x12 for File Entries)
0x01	1	Number of continuation entries following this entry
0x02	8	Time stamp
0x0A	8	Starting block number in the data area
0x12	8	Ending block number in the data area
0x1A	8	File length in bytes
0x22	30	File name in UTF-8

Table 8 - File Entry format

The entry type is used to determine that this is a File Entry and not any other type of index area entry.

The number of continuation entries is used to determine how many Continuation

Entries immediately follow an entry when the file name is too long to store in the File Entry alone. A value of zero indicates that there are no Continuation Entries. See the description of Continuation Entries for more information.

The time stamp field indicates when the File Entry information or associated file data was last modified. It is stored in the same format as all time stamps used by the Simple File System.

The starting block number and ending block number is used to find the start and end of the region within the data area used to store the file's data. The ending block number must be higher than the starting block number, unless the entry does not use any blocks in the data area (a zero length file) in which case the starting and ending block numbers should both be zero.

The file length records how long a file actually is. This value must not be larger than the starting and ending block numbers indicate, but may be significantly less to reserve space for data to be appended to a file (and will often be slightly less because most file lengths aren't a multiple of the block size).

The file name field is used to store the name of the file, including the path. For example, for the file "foo/bar.txt" this field will contain the characters "foo/bar.txt" followed by a zero to mark the end of the string (not just "bar.txt"). If the file name is too long to store in the File Entry, Continuation Entries are used. See the description of Continuation Entries for more information.

Index Area - Unusable Entry

An Unusable Entry is used to mark unusable areas of the media (for example, bad sectors), and should not be confused with Unused Entries.

Each Unusable Entry in the index area uses the following format:

Offset	Size in bytes	Description
0x00	1	Entry type (0x18 for Unusable Entries)
0x01	9	Unused/reserved
0x0A	8	Starting block number in the data area
0x12	8	Ending block number in the data area
0x1A	38	Unused/reserved

Table 9 - Unusable Entry format

The entry type is used to determine that this is an Unusable Entry and not any other type of index area entry.

The starting block number and ending block number is used to find the start and end of the region within the data area that is unusable. The ending block number must be higher than the starting block number.

Index Area - Deleted Directory Entry

Each Deleted Directory Entry in the index area uses the following format:

Offset	Size in bytes	Description
0x00	1	Entry type (0x19 for Deleted Directory Entries)
0x01	1	Number of continuation entries following this entry
0x02	8	Time stamp
0x0A	54	Directory name in UTF-8

Table 10 - Deleted Directory Entry format

The entry type is used to determine that this is a Deleted Directory Entry and not any other type of index area entry.

The time stamp field indicates when the original Directory Entry was deleted (or when the Deleted Directory Entry was created). It is stored in the same format as all time stamps used by the Simple File System.

All other fields within a Deleted Directory Entry must remain the same as the original Directory Entry to allow the directory to be restored by changing the entry type back to 0x11 (Directory Entry).

Index Area - Deleted File Entry

Each File Entry in the index area uses the following format:

Offset	Size in bytes	Description
0x00	1	Entry type (0x1A for Deleted File Entries)
0x01	1	Number of continuation entries following this entry
0x02	8	Time stamp
0x0A	8	Starting block number in the data area
0x12	8	Ending block number in the data area
0x1A	8	File length in bytes
0x22	30	File name in UTF-8

Table 11 - Deleted File Entry format

The entry type is used to determine that this is a Deleted File Entry and not any other type of index area entry.

The time stamp field indicates when the original File Entry was deleted (or when the Deleted File Entry was created). It is stored in the same format as all time stamps used by the Simple File System.

All other fields within a Deleted File Entry must remain the same as the original File Entry to allow the file to be restored by changing the entry type back to 0x12 (File Entry). This includes all blocks within the data area.

Index Area - Continuation Entry

Continuation entries are used when the name field in a preceding Directory Entry, File Entry, Deleted Directory Entry or Deleted File Entry is not long enough to contain the entire name. This forms a consecutive series of entries, beginning with a initial entry (the closest entry to the start of the media) followed by none or more continuation entries.

The format for a continuation entry is:

Offset	Size in bytes	Description
0x00	64	Entry name in UTF8

Table 12 - Continuation Entry format

To allow for much longer entry names, multiple continuation entries can be used. The number of continuation entries is stored in the associated Directory Entry, File Entry, Deleted Directory Entry or Deleted File Entry. Because this field is limited to a maximum value of 255, the number of continuation entries is limited to 255. This provides for a maximum directory name length of 16374 bytes including the zero terminator, and a maximum file name length of 16350 bytes including the zero terminator. However, it is recommended that directory names and file names be limited to a reasonable value when they are created.

Because bytes with values less than 32 aren't possible within directory names and file names, it is possible to determine if an entry is a continuation entry or any other type of entry by checking the first byte of the entry. If the first byte is below 0x20 the entry is not a continuation entry, and if the first byte is above or equal to 0x20 then it must be a continuation entry.

Time Stamps

All time stamps are signed 64 bit values that represent the number of 1/65536^{ths} of a second since the beginning of the 1st of January 1970. For example, the value 0x00000000003C0000 would represent one minute past midnight on the 1st of January 1970, while the value 0x0000000000000001 would represent roughly 15259 ns past midnight on the 1st of January 1970. All time stamps are in UTC (Universal Co-ordinated Time) so that problems with time zones and daylight savings are avoided.

This format is similar to the typical time stamp format used by Unix systems (a signed 32 bit value representing seconds since since the beginning of the 1st of January 1970), except that it is a lot more accurate (15.25 us intervals) and a lot

less limited (a range of roughly 4.46 million years before and after 1970, instead of 68 years before and after 1970). The additional accuracy can avoid problems with some utilities, and the extra range avoids the problem that most Unix based file systems will have in the year 2038.

Name Strings

All name strings used as part of the Simple File System use UTF-8 format, according to the Unicode Specification published by the Unicode Consortium.

Some characters are not allowed within a file name, directory name or volume name. These characters include all characters with codes below 32 (excluding the space character, code 0x0020), all characters with codes between 0x0080 and 0x009F (inclusive), and any of the following characters:

- " (double quote, 0x0022)
- * (asterix, 0x002A)
- : (colon, 0x003A)
- < (less than sign, 0x003C)
- > (greater than sign, 0x003E)
- ? (question mark, 0x003F)
- \ (backward slash, 0x005C)
- DEL (delete, 0x007F)
- NBSP (no break space character, 0x00A0)

For the "no break space" character (code 0x00A0), it is expected that the file system driver (or operating system code) silently replaces it with a normal space character (code 0x0020). This is intended to reduce confusion (as both characters look the same).

The forward slash character ('/', code 0x002F) must be used as a directory separator only, and is not allowed within a volume label. It is expected that some systems will silently replace these characters with a backward slash character ('\', code 0x005C) to maintain consistency with the intended environment.

All names must end with a null character (0x0000). It is also recommended that any unused bytes within any name field contain the byte 0x00.

Compatibility

The Simple File System has been carefully designed to co-exist with a legacy FAT file system. To achieve this the reserved area at the beginning of the Simple File System is made large enough to cover the entire FAT file system, with the remainder of the media used by the Simple File System.

This ability makes it possible to have a floppy disk or flash memory device (for example) with a small FAT file system that contains any instructions, drivers or files needed to access the Simple File System on the rest of the media.

Appendix A - Algorithms

[TODO - need algorithms for common operations, with the simplest possible code and most secure code (with suggestions for optimization and guidelines for things like directory entry usage and "compaction")]