

---

# SPIFFS (SPI Flash File System)

**V0.3.7**

build unknown

Copyright (c) 2013-2017 Peter Andersson (pelleplutt1976 at gmail.com)

For legal stuff, see LICENSE. Basically, you may do whatever you want with the source. Use, modify, sell, print it out, roll it and smoke it - as long as I won't be held responsible.

Love to hear feedback though!

## INTRODUCTION

Spiffs is a file system intended for SPI NOR flash devices on embedded targets.

Spiffs is designed with following characteristics in mind: - Small (embedded) targets, sparse RAM without heap - Only big areas of data (blocks) can be erased - An erase will reset all bits in block to ones - Writing pulls one to zeroes - Zeroes can only be pulled to ones by erase - Wear leveling

## BUILDING

```
mkdir build; make
```

Otherwise, configure the `builddir` variable towards the top of `makefile` as something opposed to the default `build`. Sanity check on the host via `make test` and refer to `.travis.yml` for the official in-depth testing procedure. See the wiki for integrating spiffs into projects and spiffsimg from nodemcu is a good example on the subject.

## FEATURES

What spiffs does: - Specifically designed for low ram usage - Uses statically sized ram buffers, independent of number of files - Posix-like api: open, close, read, write, seek, stat, etc - It can run on any NOR flash, not only SPI flash - theoretically also on embedded flash of a microprocessor - Multiple spiffs configurations can run on same target - and even on same SPI flash device - Implements static wear leveling - Built in file system consistency checks - Highly configurable

What spiffs does not: - Presently, spiffs does not support directories. It produces a flat structure. Creating a file with path `tmp/myfile.txt` will create a file called `tmp/myfile.txt` instead of a `myfile.txt` under directory `tmp`. - It is not a realtime stack. One write operation might last much longer than another. -

---

Poor scalability. Spiffs is intended for small memory devices - the normal sizes for SPI flashes. Going beyond ~128Mbyte is probably a bad idea. This is a side effect of the design goal to use as little ram as possible. - Presently, it does not detect or handle bad blocks. - One configuration, one binary. There's no generic spiffs binary that handles all types of configurations.

## NOTICE

0.4.0 is under construction. This is a full rewrite and will change the underlying structure. Hence, it will not be compatible with earlier versions of the filesystem. The API is the same, with minor modifications. Some config flags will be removed (as they are mandatory in 0.4.0) and some features might fall away until 0.4.1. If you have any worries or questions, it can be discussed in issue #179

## MORE INFO

See the wiki for configuring, integrating, using, and optimizing spiffs.

For design, see docs/TECH\_SPEC.

For a generic spi flash driver, see this.

## HISTORY

### 0.3.7

- fixed prevent seeking to negative offsets #158
- fixed file descriptor offsets not updated for multiple fds on same file #157
- fixed cache page not closed for removed files #156
- fixed a lseek bug when seeking exactly to end of a fully indexed first level LUT #148
- fixed wear leveling issue #145
- fixed attempt to write out of bounds in flash #130,
- set file offset when seeking over end #121 (thanks @sensslen)
- fixed seeking in virgin files #120 (thanks @sensslen)
- Optional file metadata #128 (thanks @cesanta)
- AFL testing framework #100 #143 (thanks @pjsjg)
- Testframe updates

New API functions: - [SPIFFS\\_update\\_meta](#), [SPIFFS\\_fupdate\\_meta](#) - updates metadata for a file

New config defines: - [SPIFFS\\_OBJ\\_META\\_LEN](#) - enable possibility to add extra metadata to files

---

### 0.3.6

- Fix range bug in index memory mapping #98
- Add index memory mapping #97
- Optimize SPIFFS\_read for large files #96
- Add temporal cache for opening files #95
- More robust gc #93 (thanks @dismirlian)
- Fixed a double write of same data in certain cache situations
- Fixed an open bug in READ\_ONLY builds
- File not visible in SPIFFS\_readdir #90 (thanks @benpicco-tmp)
- Cache load code cleanup #92 (thanks @niclash)
- Fixed lock/unlock asymmetry #88 #87 (thanks @JackJefferson, @dpruessner)
- Testframe updates

New API functions: - `SPIFFS_ix_map` - map index meta data to memory for a file - `SPIFFS_ix_unmap` - unmaps index meta data for a file - `SPIFFS_ix_remap` - changes file offset for index metadata map - `SPIFFS_bytes_to_ix_map_entries` - utility, get length of needed vector for given amount of bytes - `SPIFFS_ix_map_entries_to_bytes` - utility, get number of bytes a vector can represent given length

New config defines: - `SPIFFS_IX_MAP` - enable possibility to map index meta data to memory for reading faster - `SPIFFS_TEMPORAL_FD_CACHE` - enable temporal cache for opening files faster - `SPIFFS_TEMPORAL_CACHE_HIT_SCORE` - for tuning the temporal cache

### 0.3.5

- Fixed a bug in fs check
- API returns actual error codes #84) (thanks @Nails)
- Fix compiler warnings for non-gcc #83 #81 (thanks @Nails)
- Unable to recover from full fs #82 (thanks @rojer)
- Define SPIFFS\_O\_\* flags #80
- Problem with long filenames #79 (thanks @psjg)
- Duplicate file name bug fix #74 (thanks @igrr)
- SPIFFS\_eof and SPIFFS\_tell return wrong value #72 (thanks @ArtemPisarenko)
- Bunch of testframe updates #77 #78 #86 (thanks @dpreussner, @psjg a.o)

### 0.3.4

- Added user callback file func.

- 
- Fixed a stat bug with obj id.
  - SPIFFS\_probe\_fs added
  - Add possibility to compile a read-only version of spiffs
  - Make magic dependent on fs length, if needed (see #59 & #66) (thanks @hreintke)
  - Exposed SPIFFS\_open\_by\_page\_function
  - Zero-size file cannot be seek #57 (thanks @lishen2)
  - Add tell and eof functions #54 (thanks @raburton)
  - Make api string params const #53 (thanks @raburton)
  - Preserve user\_data during mount() #51 (thanks @rojer)

New API functions: - [SPIFFS\\_set\\_file\\_callback\\_func](#) - register a callback informing about file events - [SPIFFS\\_probe\\_fs](#) - probe a spi flash trying to figure out size of fs - [SPIFFS\\_open\\_by\\_page](#) - open a file by page index - [SPIFFS\\_eof](#) - checks if end of file is reached - [SPIFFS\\_tell](#) - returns current file offset

New config defines: - [SPIFFS\\_READ\\_ONLY](#) - [SPIFFS\\_USE\\_MAGIC\\_LENGTH](#)

### 0.3.3

**Might not be compatible with 0.3.2 structures. See issue #40** - Possibility to add integer offset to file handles - Truncate function presumes too few free pages #49 - Bug in truncate function #48 (thanks @PawelDefee) - Update spiffs\_gc.c - remove unnecessary parameter (thanks @PawelDefee) - Update INTEGRATION docs (thanks @PawelDefee) - Fix pointer truncation in 64-bit platforms (thanks @igrr) - Zero-sized files cannot be read #44 (thanks @rojer) - (More) correct calculation of max\_id in obj\_lu\_find #42 #41 (thanks @lishen2) - Check correct error code in obj\_lu\_find\_free #41 (thanks @lishen2) - Moar comments for SPIFFS\_lseek (thanks @igrr) - Fixed padding in spiffs\_page\_object\_ix #40 (thanks @jmattsson @lishen2) - Fixed gc\_quick test (thanks @jmattsson) - Add SPIFFS\_EXCL flag #36 - SPIFFS\_close may fail silently if cache is enabled #37 - User data in callbacks #34 - Ignoring SINGLETON build in cache setup (thanks Luca) - Compilation error fixed #32 (thanks @chotasanjiv) - Align cand\_scores (thanks @hefloryd) - Fix build warnings when SPIFFS\_CACHE is 0 (thanks @ajaybhargav)

New config defines: - [SPIFFS\\_FILEHDL\\_OFFSET](#)

### 0.3.2

- Limit cache size if too much cache is given (thanks pgeiem)
- New feature - Controlled erase. #23
- SPIFFS\_rename leaks file descriptors #28 (thanks benpicco)

- 
- moved dbg print defines in test framework to params\_test.h
  - lseek should return the resulting offset (thanks hefloryd)
  - fixed type on dbg ifdefs
  - silence warning about signed/unsigned comparison when spiffs\_obj\_id is 32 bit (thanks ben-picco)
  - Possible error in test\_spiffs.c #21 (thanks yihcdaso-yeskela)
  - Cache might writethrough too often #16
  - even moar testrunner updates
  - Test framework update and some added tests
  - Some thoughts for next gen
  - Test sigsevs when having too many sectors #13 (thanks alonewolfx2)
  - GC might be suboptimal #11
  - Fix eternal readdir when objheader at last block, last entry

New API functions: - `SPIFFS_gc_quick` - call a nonintrusive gc - `SPIFFS_gc` - call a full-scale intrusive gc

### 0.3.1

- Removed two return warnings, was too triggerhappy on release

### 0.3.0

- Added existing namecheck when creating files
- Lots of static analysis bugs #6
- Added rename func
- Fix SPIFFS\_read length when reading beyond file size
- Added reading beyond file length testcase
- Made build a bit more configurable
- Changed name in spiffs from “errno” to “err\_code” due to conflicts compiling in mingw
- Improved GC checks, fixed an append bug, more robust truncate for very special case
- GC checks preempts GC, truncate even less picky
- Struct alignment needed for some targets, define in spiffs config #10
- Spiffs filesystem magic, definable in config

New config defines: - `SPIFFS_USE_MAGIC` - enable or disable magic check upon mount - `SPIFFS_ALIGNED_OBJECT_INDEX_TABLES` - alignment for certain targets

---

New API functions: - `SPIFFS_rename` - rename files - `SPIFFS_clearerr` - clears last errno  
- `SPIFFS_info` - returns info on used and total bytes in fs - `SPIFFS_format` - formats the  
filesystem - `SPIFFS_mounted` - checks if filesystem is mounted