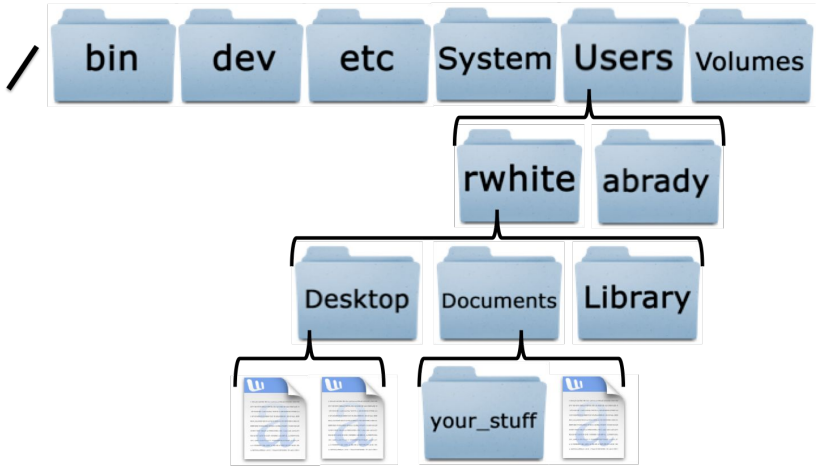
TEFS: A Flash File System for Use on Memory Constrained Devices

Wade Penson

What Is a File System?

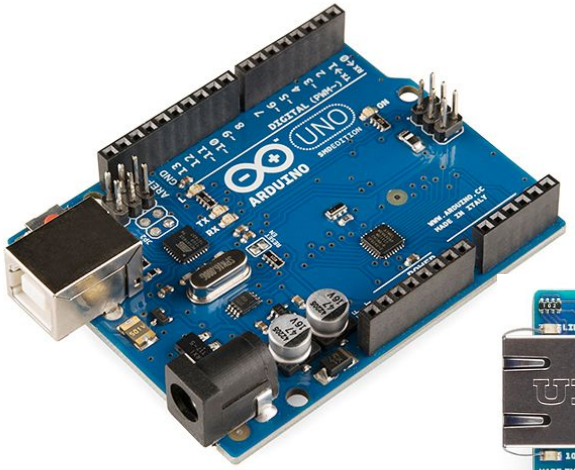
A file system keeps track of files on a storage device. It is made up of data structures and methods to accomplish this.



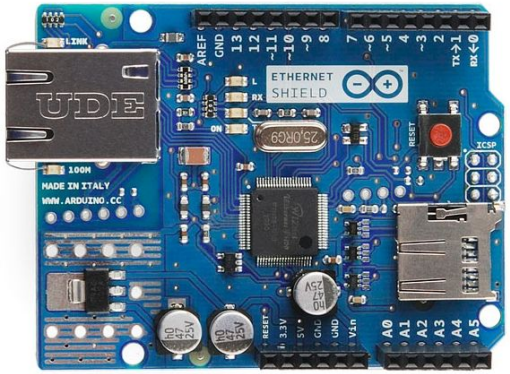
Source: crashwhite.com

What Do I Mean by Memory Constrained Devices?

	Arduino UNO	**iPhone 6s**
Flash Memory	32 KB	16,000,000 KB Minimum
SRAM	2 KB	2,000,000 KB
Clock Speed	16 MHz	1,840 MHz



Source: Wikimedia



Source: Arduino

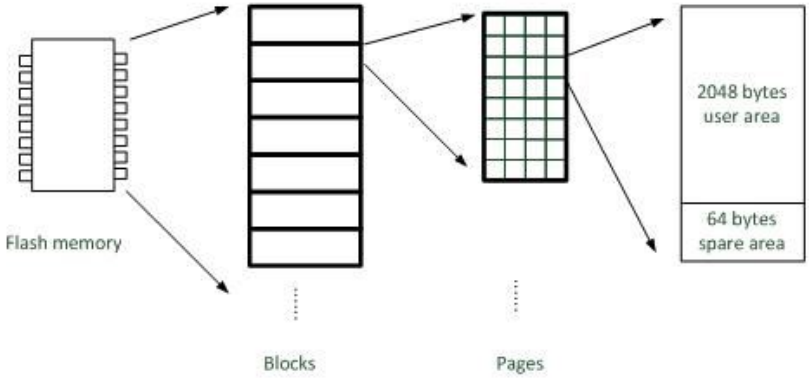
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas eget velit erat. Nullam velit felis, consequat non vulputate ut, sodales bibendum velit. Phasellus a mi et enim blandit ullamcorper sollicitudin sit amet metus. Suspendisse bibendum vehicula ultrices. Sed laoreet orci ac lobortis imperdiet. Morbi quis lobortis turpis. Vestibulum auctor mollis sapien, maximus laoreet quam porta nec. Nam eleifend, enim ac consectetur condimentum, justo neque imperdiet nulla, eget luctus dolor tellus eget diam. Quisque ornare aliquet quam, pellentesque mollis est scelerisque sit amet. Aenean ornare fringilla consequat. Morbi non hendrerit risus. Duis iaculis nisi augue, non sollicitudin elit scelerisque nec. Integer lacus mi, gravida id velit nec, pharetra faucibus velit. In nec enim in neque egestas finibus in a orci. Maecenas vitae egestas ipsum. Aliquam dignissim orci laoreet, pulvinar velit in, condimentum tellus. In hac habitasse platea dictumst. Maecenas a odio sed nunc volutpat interdum id non metus. Integer quis ante aliquet, eleifend arcu sit amet, faucibus elit. Donec blandit leo nulla, ut feugiat mauris tristique a. Cras vel dui sed lectus tempor vestibulum sit amet quis libero. Duis mollis ligula vel ante tincidunt, a sodales felis pharetra. Donec dictum nunc cursus, commodo eros ac, volutpat massa. Aenean eget orci sed nulla molestie posuere. Ut feugiat eu quam quis scelerisque. Nullam odio arcu, posuere in velit a, vehicula lobortis risus. In quis risus id quam semper pharetra. Aenean eu libero sem. In hac habitasse platea dictumst. Proin tristique arcu ut magna mollis, non maximus est suscipit. Curabitur scelerisque elit eget urna vestibulum, nec iaculis ante sodales. Integer scelerisque turpis metus. In mollis tempor cursus. Sed efficitur lacus sit amet ipsum feugiat, vitae tincidunt orci dapibus. Praesent vestibulum a odio nec mattis. Donec at nisi sit amet turpis commodo sodales. Proin ut lectus bibendum, dictum augue in, viverra nibh. Etiam ultricies nibht augue, quis placerat urna fringilla nullam.

Types of Persistent Memory

- TEFS is designed for flash memory.
- The most common flash memory is NAND and NOR flash.
- Flash memory is capable of fast random reads and writes as compared to a hard disk drive.

Memory Characteristics of Flash Memory

- Flash memory is comprised of blocks and blocks are comprised of pages.
- Pages are the smallest accessible unit in NAND flash for reading and writing.
- Clusters are groups of pages.
- SD cards and USB sticks are NAND flash.
  - They have a Flash Translation Layer
  - You read from and write to logical pages

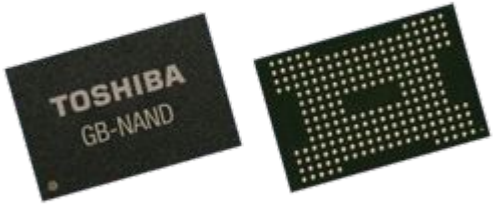


Types of Persistent Memory

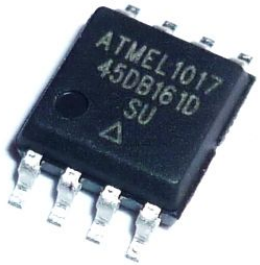
Source: Amazon.ca

Source: McNeese State University

Source: wme.my



Source: Toshiba



Source: sm-elektronik.pl

Why Develop Another File System?

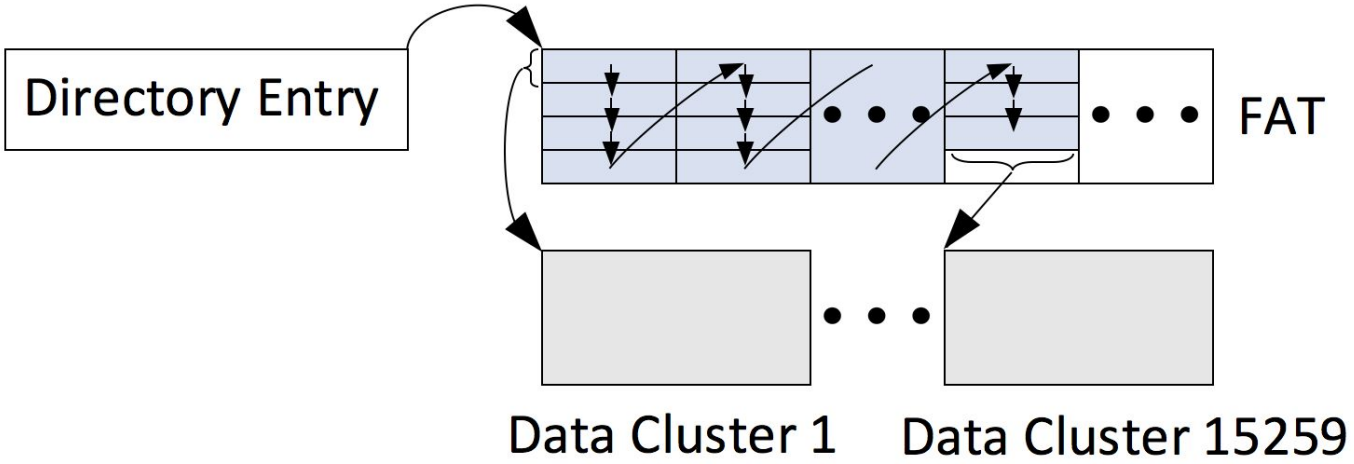
- Most file systems for microcontrollers that use flash memory with an FTL are FAT or a derivative of FAT.
  - Despite FAT's simplicity as a file system, there are performance issues regarding random reads and writes in large files.
- IonDB is a key-value store for embedded devices which uses various data structures that store data in persistent memory. One of them is a B+ Tree which utilizes random reading and writing.

- Lwext4 is a port of the ext2,3,4 file system that works on embedded devices.
  - It consumes 8KB of RAM at minimum.

Introduction to FAT

Layout of FAT32

File Allocation Table



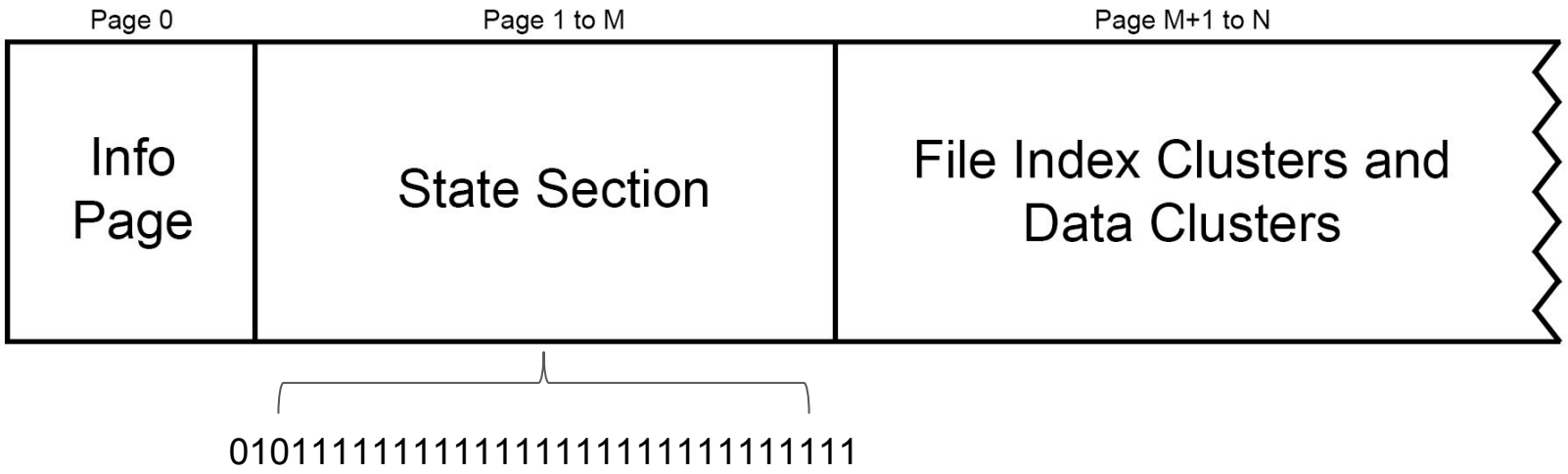
Directory Structure of FAT32

The directory structure manages the metadata for a file and the lookup of files by file names.

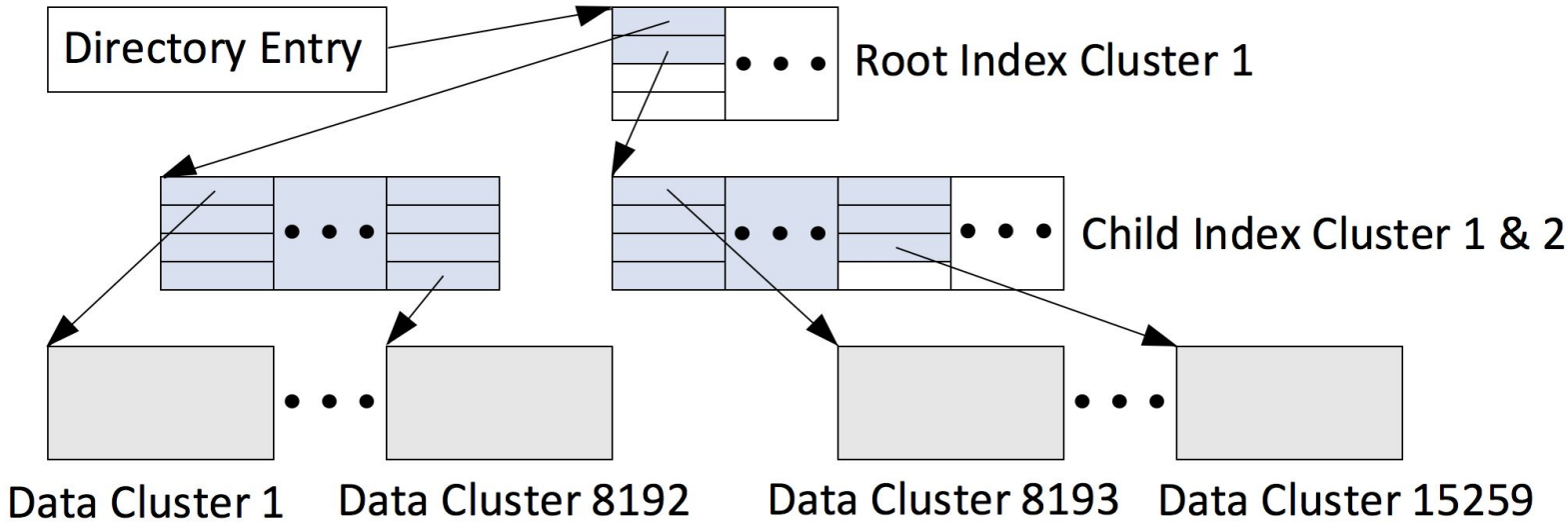
- The root directory is a file or a linked list of data clusters.
  - Extra directories are separate files.
- A directory has an array of directory entries that are 32 bytes each.

Introduction to TEFS

Layout of TEFS



File Index Structure

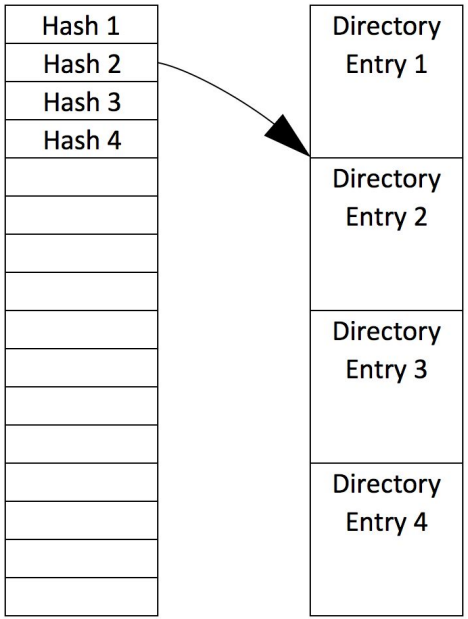


Directory Structure

The directory structure is a linear search like FAT32's directory structure but with some improvements.

Two files make up the directory structure:

- Hash Entries File
- Metadata File



FAT File System vs. TEFS

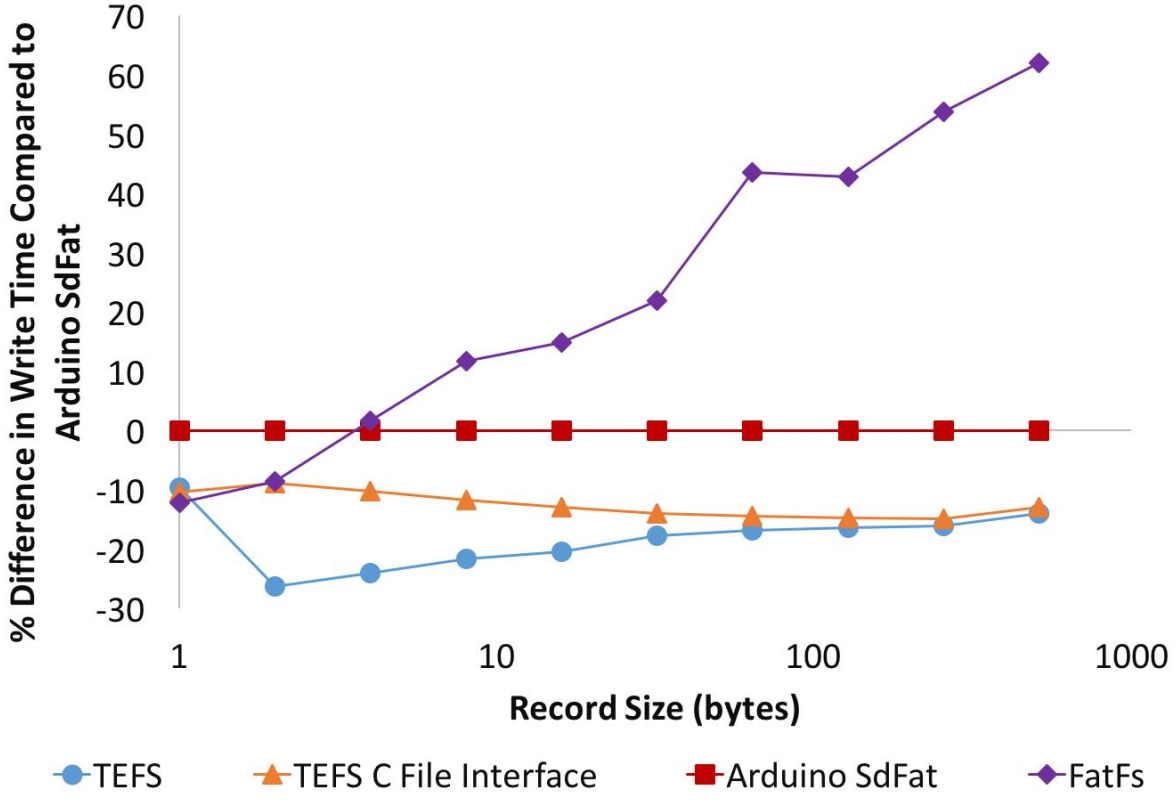
Benchmarks

- The benchmarks were done on an Arduino Uno with a 16GB UHS 1 Sandisk MicroSD card.
- Two popular FAT libraries for microcontrollers were compared against - the Arduino SdFat and FatFs libraries.
- The cluster size and page size were set to be the same for FAT and TEFS.
- Time benchmarks were an average of 5 runs.

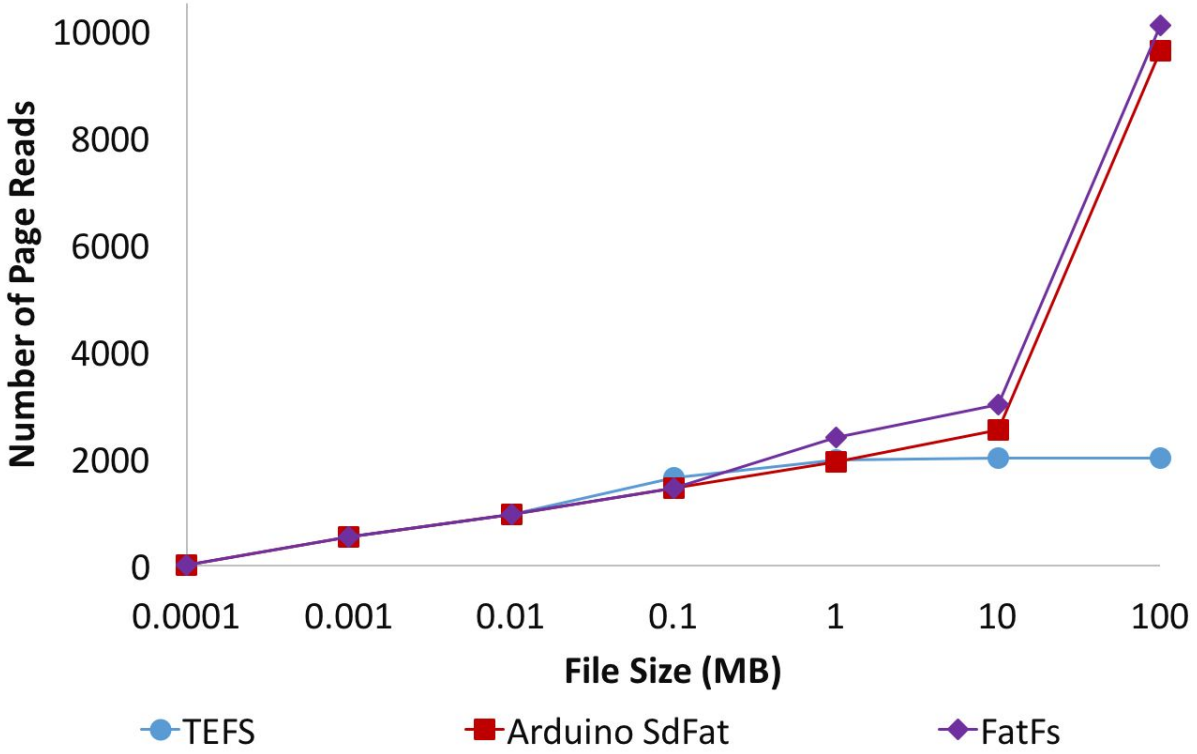
Sequentially Writing 1000 Pages

File System	**Number of Page Reads (< 511 bytes, 512 bytes)**	**Number of Page Writes (< 511 bytes, 512 bytes)**
TEFS	31 / 31	1030 / 1030
Arduino SdFat	17 / 2	1017 / 1003
FatFs	17 / 2	1019 / 1005

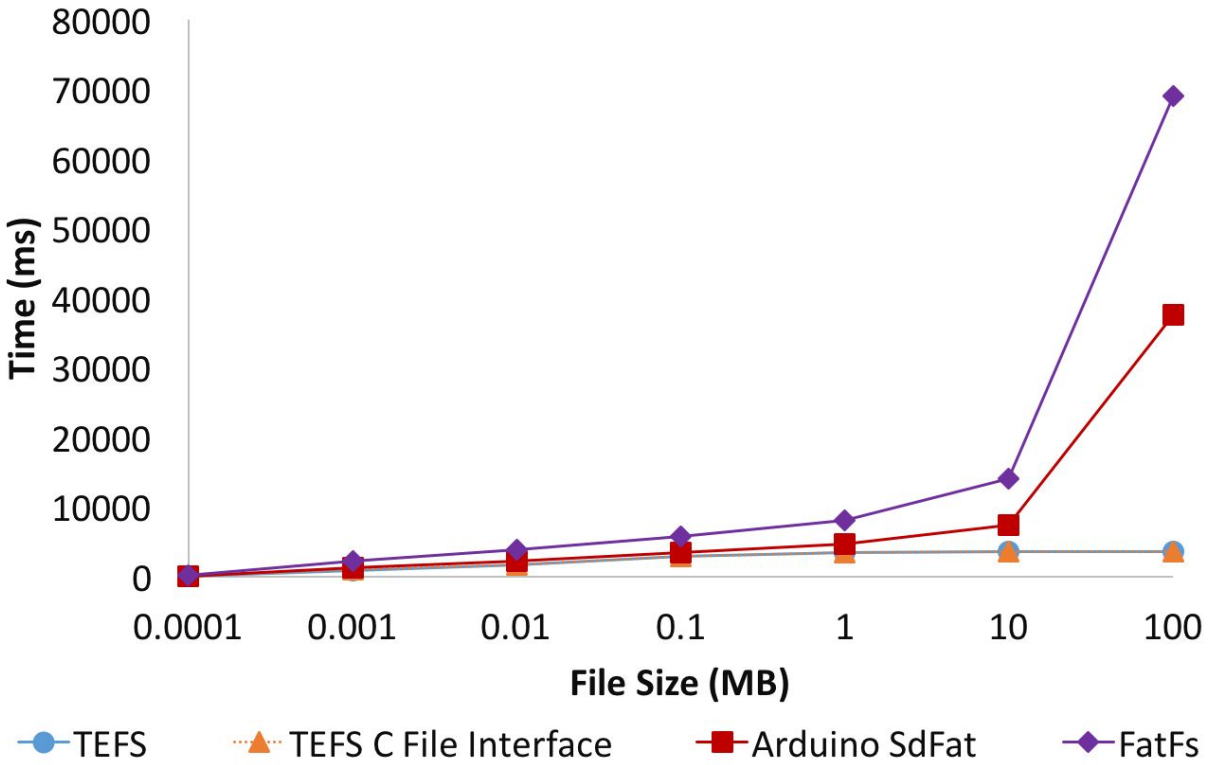
Sequentially Writing 1000 Pages



Reading or Writing 1000 Bytes at Random Locations



Reading or Writing 1000 Bytes at Random Locations



File Creation (and Removal)

	Number of Page Reads				Number of Page Writes			
File System	**1 File**	**10 Files**	**100 Files**	**1000 Files**	**1 Files**	**10 Files**	**100 Files**	**1000 Files**
TEFS v1	7	52	766	35754	6	51	501	5001
TEFS v2	9	63	597	9350	6	60	600	6000
Arduino SdFat	0	0	71	6359	1	2	6	189

Library Sizes

File System	Text Size (bytes)	Dynamic Memory (bytes)	Memory per File (bytes)
TEFS	10364	647	34
TEFS C File Interface	12260	683	41
Arduino SdFat	14752	608	27
FatFs	14879	584	36

Trade-offs

TEFS	FAT
Extra clusters are needed for each file and the state section is also needed	The File Allocation Table is a compact data structure but is allocated during formatting
Scanning for a file is faster for large files but there are more page reads and writes when creating or removing a file	Creating or removing a file is fast but scanning for a file is slow if there are many files
Slightly faster sequential reads and writes	Slightly slower sequential reads and writes
Significantly faster random reads and writes for large files	For small files, random reads and writes are comparable but are slow for large files
Only has a root directory	Has multiple directories (or folders)

Trade-offs

TEFS	FAT
Supports custom file name sizes and metadata	Can support long file names if implementation supports it
Supports custom metadata set by the user	No custom metadata
Max file size depends on page size and cluster size but it can be quite large	FAT32 has a 4GB file size limit
Code size is much smaller but uses slightly more RAM	Larger code size but uses slightly less RAM as seen in the tested implementations
Cannot view or extract files on Windows, Linux, etc.	Supported by all major operating systems

Conclusion

In summary, FAT performs better when the are fewer files, smaller files, and files are created and removed frequently. TEFS, however, performs better when there are more files, larger files, and files are opened and closed often but not created and removed as often.

Future improvements of TEFS could include supporting major operating systems to allow formatting and managing files.

Acknowledgement

I would like to thank NSERC for supporting my research on this project during the summer of 2015.

NSERC CRSNG People. Discovery. Innovation.

If you would like to find out more about TEFS, check out my IEEE Canadian Conference publication.

IEEE Canada

Max File Size

A device that has a page size of 512 bytes, 4 byte addresses, and a cluster size of 32KiB has a max file size of 2TiB.

TEFS C File Interface

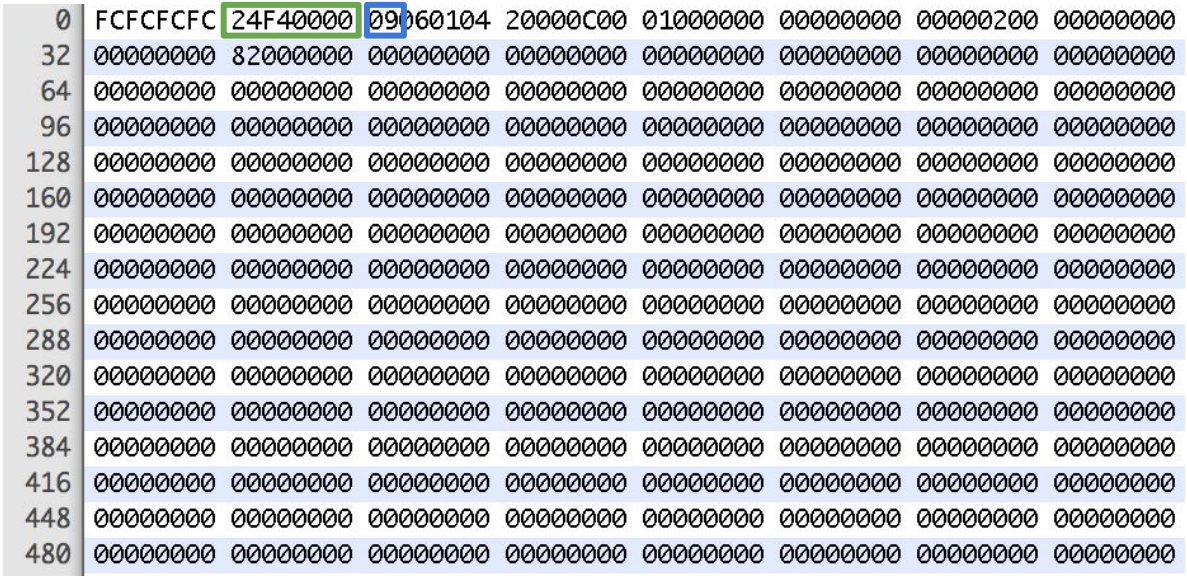
T_FILE *t_fopen(char *file_name, char *mode);

size_t t_fwrite(void *ptr, size_t size, size_t count, T_FILE *fp);

int8_t t_remove(char *file_name);

Information Page

The information page is similar to the boot sector for FAT. It contains the details from formatting that the file system requires to function.



0	FCFCFCFC	24F40000	09060104	20000C00	01000000	00000000	00000200	00000000
32	00000000	82000000	00000000	00000000	00000000	00000000	00000000	00000000
64	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
96	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
128	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
160	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
192	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
224	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
256	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
288	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
320	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
352	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
384	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
416	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
448	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
480	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

Operations of TEFS

- Formatting
- Opening, Closing, and Removing files
- Reading from and Writing to files

File Open

	Number of Page Reads			
File System	**1 File**	**10 Files**	**100 Files**	**1000 Files**
TEFS v1	2	2	8	64
TEFS v2	3	3	3	10