Embedded Linux Conference
April 11- 13, 2011
San Francisco , CA

Texas Instruments
Linux Development Center

Power Fail Safe FAT File System

The FAT 12/16/32 file systems

- Logging mechanism
- Recovery
- FAT Specification compatibility

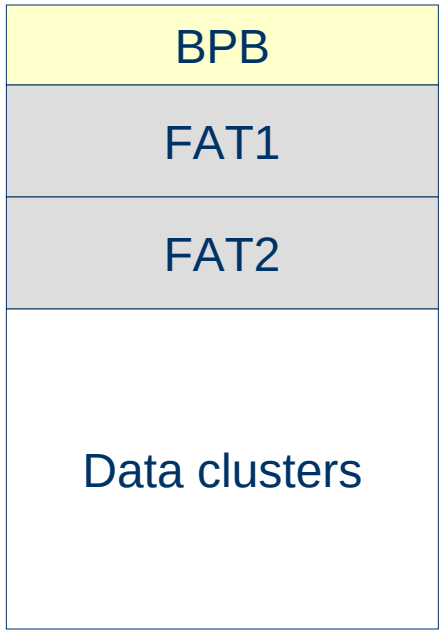
Keshava Munegowda
Texas Instruments (India) Pvt Ltd
Bangalore

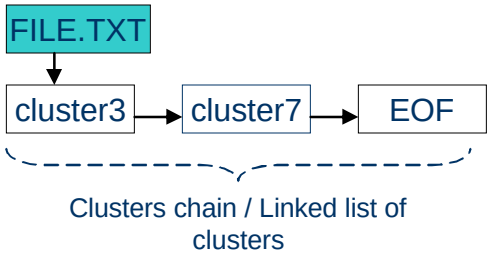
Agenda

- FAT file system
- Need of Power fail safe FAT
- State of Art
- TFAT
- KFAT, RFS and TFS4
- TI-LFAT: TI Log based Power fail safe FAT
- Directory entry Logging
- FAT entry Logging
- Log Record
- File system operations with Logging and Committing
- References

Texas Instruments
Linux Development Center

FAT File system





BPB - BIOS Parameter Block

- BIOS : Basic Input-Output System
- Also Called as “Boot Sector”
- Specifies
  - Number of sectors in the storage partition/disk/device
  - Number of FATs (File Allocation Table)
  - Sectors per cluster

FAT1 – File Allocation Table

- Linear linking (chain) of data clusters of the file/directory

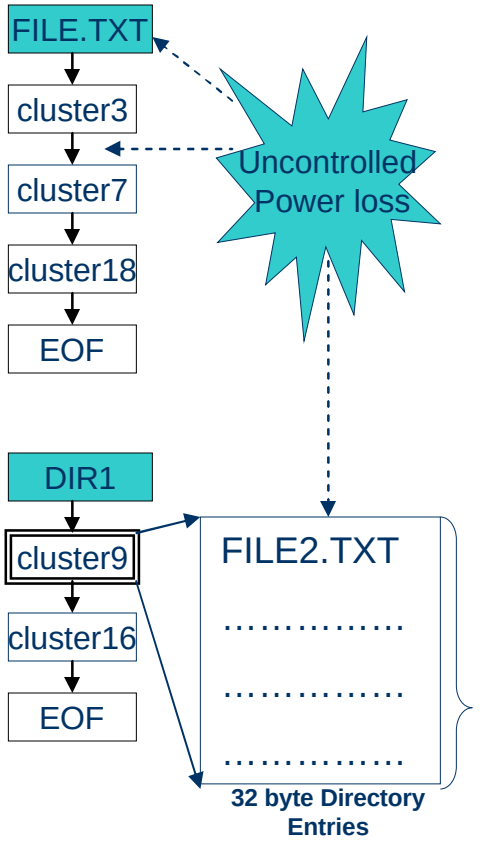
FAT2 – Backup of FAT1

Data clusters

- Group of physical/logical sectors/blocks
- Contains directories or Files data

Texas Instruments
Linux Development Center

Need of Power fail safe FAT



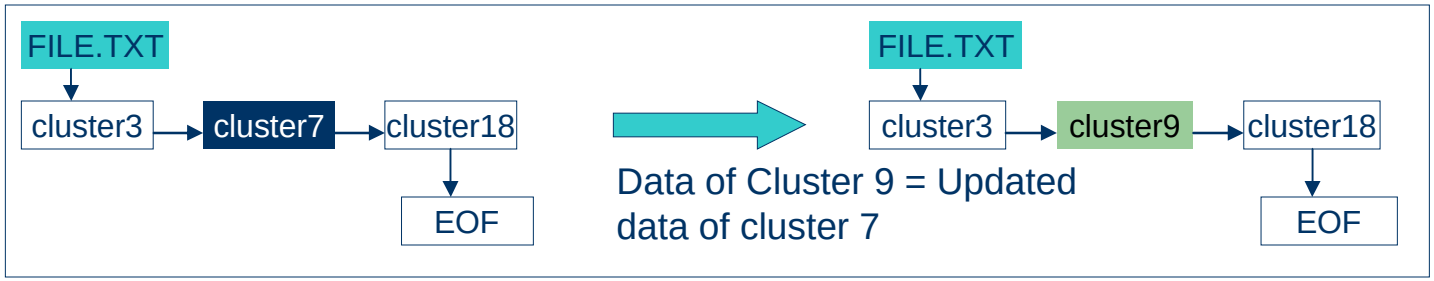
- FAT File system is not power fail safe.
- This means during file or directory update if there is uncontrolled power loss, then it can causes incorrect file system update.
- Files are data of the directories; Typically the File/directory information such as file/directory name, extension, attribute, size, and starting data cluster number are stored in the form of 32 Byte Directory Entries.
- Meta Data of the file/directory means
  - FAT entries specify the cluster chain of the file/directory
  - 32 Byte Directory Entry
- Incorrect Update of File system Meta data corrupts the organization of files/directories and hence creates garbage data.
- File system format is typical solution to re use the storage device; but it is not recovery.
- In FAT, the update of FAT entries and directory entries should be logged or should be atomic operation.

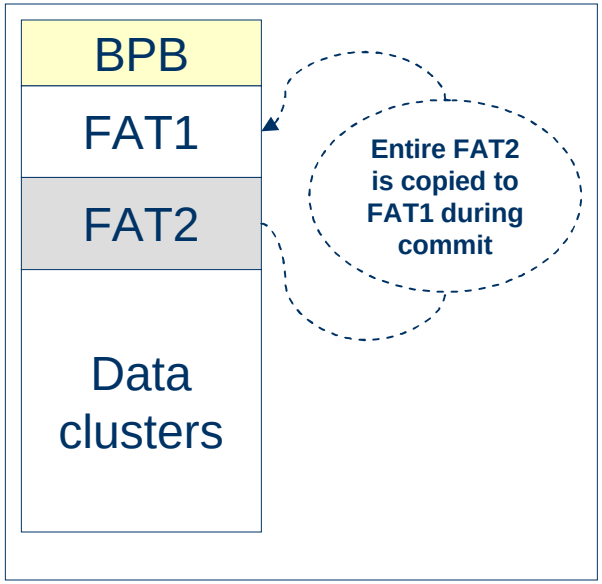
State of Art

- TFAT
  - Transaction-safe FAT File system, Microsoft.
- KFAT
  - Log based Transactional FAT file system for embedded mobile systems, Samsung.
- RFS
  - Robust FAT File system, Samsung
- TFS4
  - Transactional File system 4 for oneNAND flash memory, Samsung

Texas Instruments
Linux Development Center

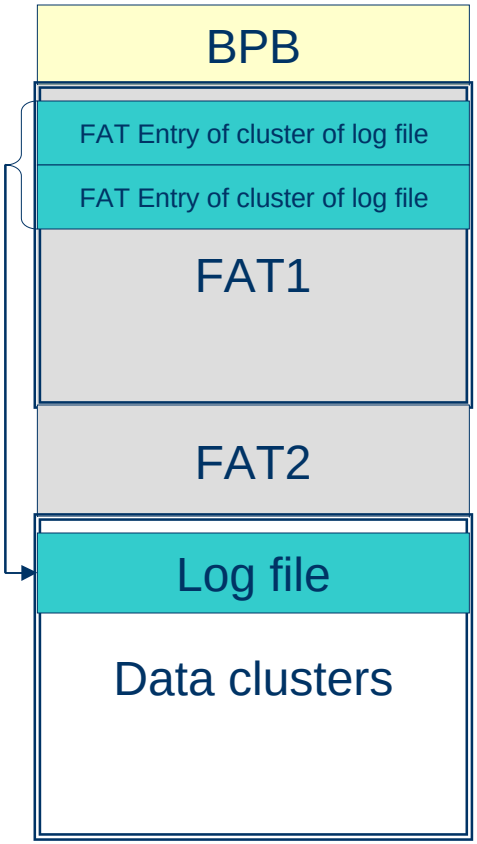
TFAT





- Developed by Microsoft; Available in Windows Embedded CE 6.0.
- Typically FAT2 is set as active working FAT.
- All FAT entries are first updated in FAT2 and then entire FAT2 is copied to FAT1 during commit.
- File/directory update is performed by allocating new data cluster and then write updated data instead of updating existing data cluster. This causes additional writes to FAT entries.
- The Root directory updates of FAT12 and FAT16 are not transactions safe. This is because Root directory in these file systems exists in a fixed location.
- Performance is lower

KFAT, RFS and TFS4

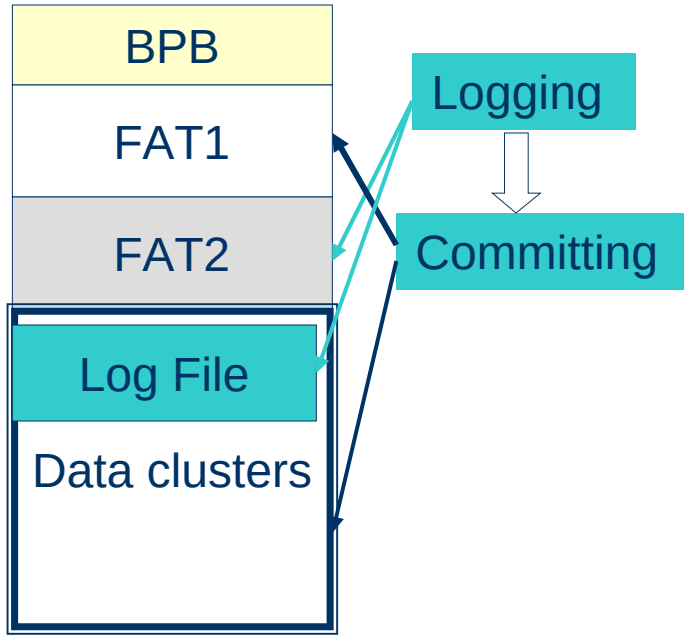


Structure of a Log file

- Log based Transactional FAT file system for embedded mobile systems by M.S. Kwon, S.H. Bae, S.S. Jung, D.Y. Seo, and C.K. Kim, in Proceedings of 2005 US-Korea Conference, ICTS-142,2005
- All file system updates such as FAT entry updates and 32 byte directory/file are logged.
- During commit operation, the logged entries are processed and finally FAT1, FAT2 and directory entries are updated.
- Typically, The log file is the first file in the root directory.
- The Log file size is fixed
- Contains 13 different Log types
- But, definitions of all log types are not known.
- RFS and TFS4 are based on KFAT.
- TFS4 is specific to Samsung's oneNAND flash memory.

Texas Instruments
Linux Development Center

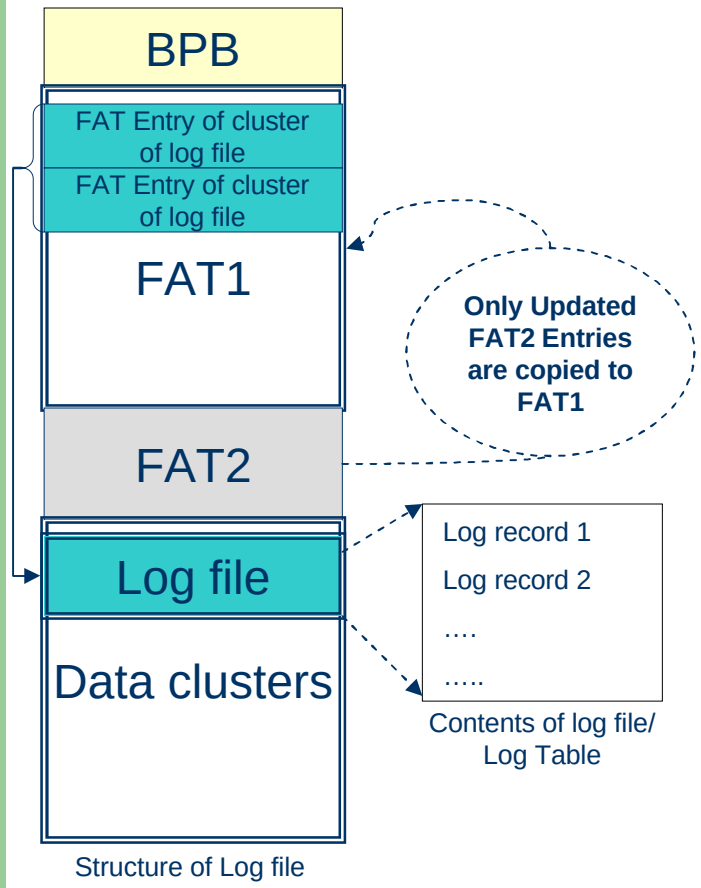
TI-LFAT : TI Log based power fail safe FAT file system



- Defines
  - **FAT Entries Logging**
    - Similar to TFAT; But Entire FAT2 is not completely copied to FAT1. Only Updated FAT2 entries are copied to FAT1.
  - **Directory Entries Logging**
    - Similar to KFAT; But, FAT entries Updates are not logged in to Log file, only 32 Byte Directory entries updates are logged.
- Logging
  - Log file and FAT2 Update
- Commit
  - FAT1 and Data clusters Update
- Objectives of TI-LFAT:
  - Secure the FAT entries and 32 Byte Directory Entry Updates
  - Minimize the Reduction of performance
  - Maintain the FAT specification Compatibility
    - first updating the FAT2 entries and then copying updated entries to FAT1 does not break the compatibility.
    - Log file create/update is generic file update and does not break the compatibility.
    - Windows PC shows log file as a file existing in root directory without any errors.

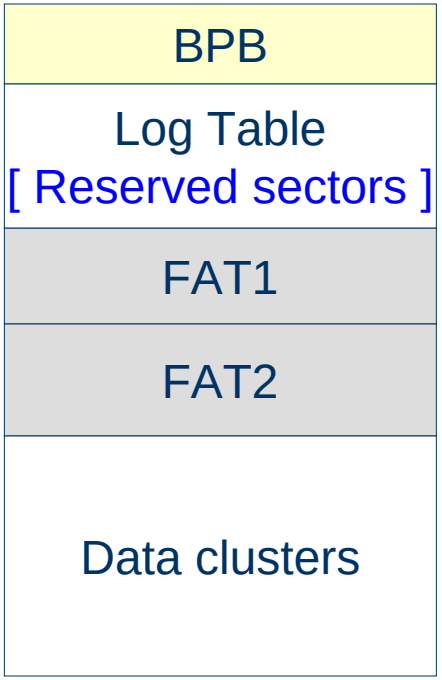
Texas Instruments
Linux Development Center

TI-LFAT : TI Log based power fail safe FAT file system



- Defines “Log Records” for Directory Entry Logging.
- Log records contains the Updated information of the FAT 32 Byte directory Entries
- The FAT entries are always updated along with 32 byte directory entry update
- This TI-LFAT defines Log Table.
- The Log table serves the same purpose of Log file and it is secure from accidental user updates.
- The Implementation can choose either log file or log table.
- The 32 byte directory entry updates are always first appended to log file or log table and then updated in the file system.
- The FAT entries are first updated in FAT2 and then only updated entries are copied to FAT1.
- This TI-LFAT combines FAT entries update techniques described TFAT, but not the complete FAT copy mechanism, and usage of log file is as described in KFAT with improved performance by not logging FAT entries Updates and reduced the logging space.

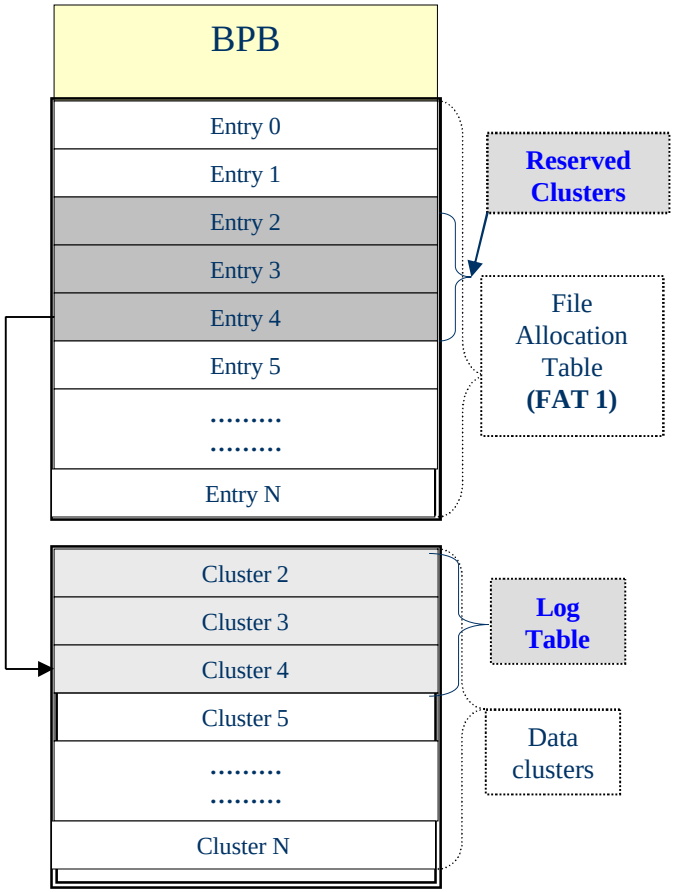
Directory entry Logging



- Log file/Table is contains the log records
- The log records specifies file system operation
- The log records can be placed in
  - The log file
  - The log table in reserved clusters
  - The log table in reserved sectors
- Log file is a fixed size file in the Root Directory.
- Log records can be placed in a set of reserved sectors.
- The reserved sectors are always exists in between BPB and FAT1.
- Typically the number of reserved sectors are configured during file system format.
- The file system does not allocate the reserved sectors to any file/directory to store the data of file/directory.
- The log table placed in reserved sectors does not break the FAT file system specification compatibility. This means , The Windows PC or Linux PC doest not access (read/write) these reserved sectors and does not shows any errors about the existence of the reserved sectors.
- These reserved sectors are invisible to user and file system applications. This means, these clusters are not part of any file and directory and hence user/file system applications can not access data of these clusters.

Texas Instruments
Linux Development Center

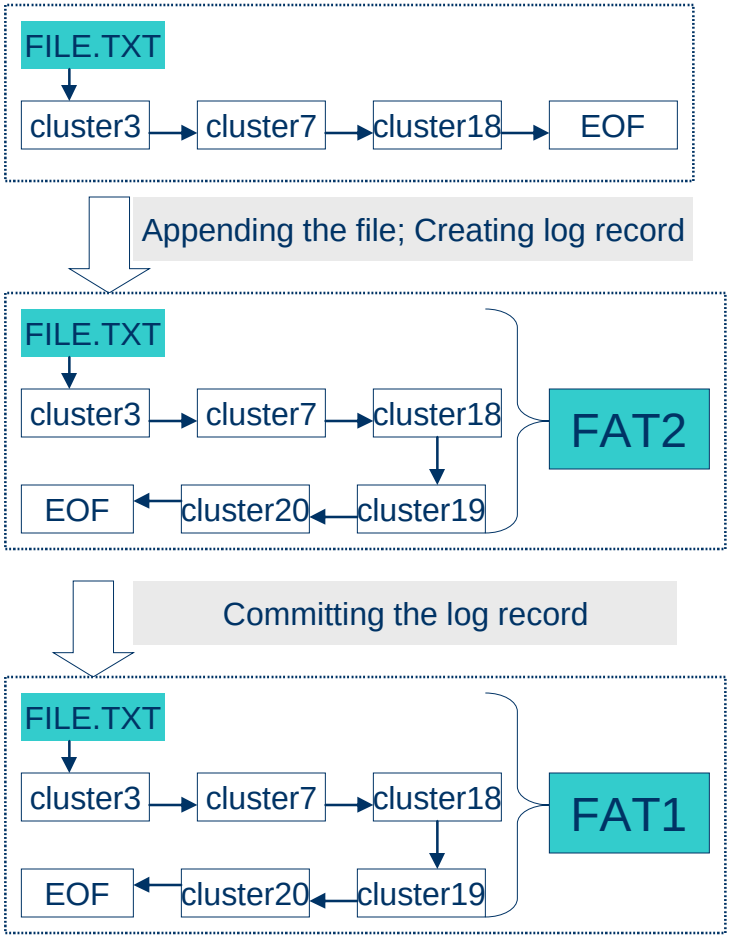
Directory entry Logging cont...



- The log table can be placed in reserved clusters.
- The Reserved clusters are created by writing the 28 bit value 0x0FFFFFF6 to required FAT entries for FAT32 file system.
- The 16bit value 0xFFF6 is written to required FAT entries in case of FAT16 file system to create reserved clusters.
- The 12 bit value 0xFF6 is written to required FAT entries in case of FAT16 file system to create reserved clusters.
- The file system does not allocate the reserved sectors to any file/directory to store the data of file/directory.
- The log table placed in reserved clusters does not break the FAT file system compatibility. This means, The Windows PC or Linux PC or any other FAT implementation adhering to FAT specification standard doest not access (read/write) these reserved clusters and does not shows any errors about the existence of the reserved clusters.
- The log file is visible to user as file in root directory where as log table in reserved clusters is invisible to user and file system applications. This means, these clusters are not part of any file and directory and hence user/file system applications can not access data of these clusters.
- The Advantage of having reserved clusters as Log Table is, the allocated reserved clusters can be re used for data storage whenever reserved clusters are not required; But in case of reserved sectors, this is not possible and only by file system format operation number of reserved sectors can be reduced.

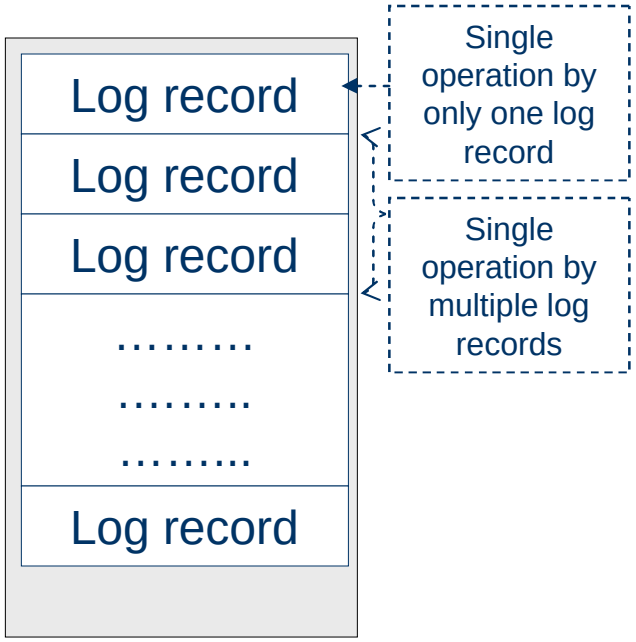
Texas Instruments
Linux Development Center

FAT entry Logging



- The FAT entries are first updated in FAT2 and then during committing log records of the log table/file, only the updated entries are copied to FAT1.
- FAT1 is always the good FAT. Where are FAT2 is always the Dirty FAT.
- TFAT typically copies entire FAT2 to FAT1, where as in this TI-LFAT File system
  - While processing the log records of the log table/file, the starting FAT cluster entries are identified.
  - The FAT entries chain from starting FAT cluster entries are traversed in FAT2 and same entries are modified in FAT1.
  - In the example,
    - The cluster 19 and cluster 20 are appended to file FILE.TXT; First the log record is created for the file update and then the fat entries are updated in FAT1.
    - After the log record created and file data written to clusters 19 and 20; the log record is committed to file system by updating the directory entry of the FILE.TXT and copying the updated cluster entries numbered 19 and 20 to FAT1.

Log Record



Contents of Log File/Log Table

- Log record is similar to FAT 32 Byte directory entry.
- A Single file system operation is specified by the single or multiple Log records.
- Each of the following operations requires single log record to specify
  - Creation with SFN (Short File Name)
  - Deletion
  - Update
  - Truncation
- Each of the following operations requires Two log record to specify
  - Creation with LFN (Long File Name)
  - Rename

Texas Instruments
Linux Development Center

Log Record Structure

FAT 32 byte Directory Entry Structure

Name	Offset (byte)	Size (bytes)	Description
DIR_Name	0	11	Short name.
DIR_Attr	11	1	File attributes:
DIR_NTRes	12	1	Reserved
DIR_CrtTimeTenth	13	1	Millisecond stamp at file creation time
DIR_CrtTime	14	2	Time file was created.
DIR_CrtDate	16	2	Date file was created.
DIR_LstAccDate	18	2	Last access date.
DIR_FstClusHI	20	2	High word of this entry's first cluster number
DIR_WrtTime	22	2	Time of last write.
DIR_WrtDate	24	2	Date of last write.
DIR_FstClusLO	26	2	Low word of this entry's first cluster number.
DIR_FileSize	28	4	file size in bytes.

Log Record Structure

Name	Offset (bytes)	Size (bytes)	Description
DIR_Name	0	11	Short name.
DIR_Attr	11	1	File attributes
DIR_LFNCount	**12**	**1**	**Long File names Entries Count**
DIR_Log	**13**	**1**	**Logging operation; 0x01 – Create; 0x02 – Update; 0x03 – Delete; 0x04 - Truncate; 0x05 –Rename; 0x00 – DIR_log operation of previous Log record.**
DIR_DCluster	**14**	**4**	**Directory cluster; Cluster in which this 32 byte directory entry exist.**
DIR_Index	**18**	**2**	**Index in the DIR_Dcluster;This index points to 32 byte directory entry**
DIR_FstClusHI / **DIR_LstClusHI**	20	2	High word of this entry's first cluster number **If DIR_log is Update (0x03) then this value indicates high word of last cluster before update**
DIR_Cluster	**22**	**4**	**If DI_log is Delete (0x03) or Rename (0x5) then DIR_cluster is the cluster prior to DIR_DCluster; the index at this cluster value in FAT contains DIR_DCluster If DIR_log is Update (0x03) then DIR_Cluster is starting cluster in the FAT entry from which the cluster chain needs to be copied from FAT2 to FAT1. If DIR_log is truncate (0x04) then DIR_cluster is starting cluster from which the cluster chain deletion should start**
DIR_FstClusLO/ **DIR_LstClusLO**	26	2	Low word of this entry's first cluster number. **If DIR_log is Update (0x03) then this value indicates low word of last cluster before update**
DIR_FileSize	28	4	file size in bytes.

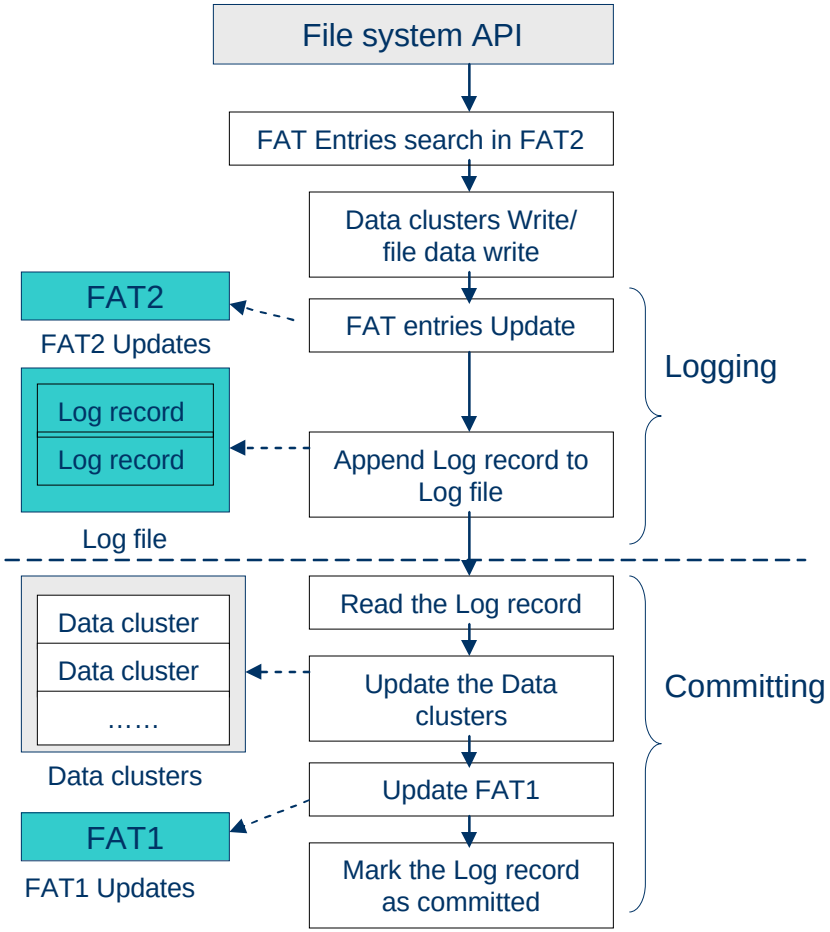
Log Record with LFN

32 Bytes

DIR_name	DIR_attr	DIR_LFNCount = 9	DIR_Log = 1	…….
1st 32 bytes of Unicode Long File name characters (16 Unicode characters)				
2nd 32 bytes of Unicode Long File name characters (16 Unicode characters)				
…….. ……..				
9th (Last) 32 bytes or less than 32 bytes of Unicode Long File name characters				

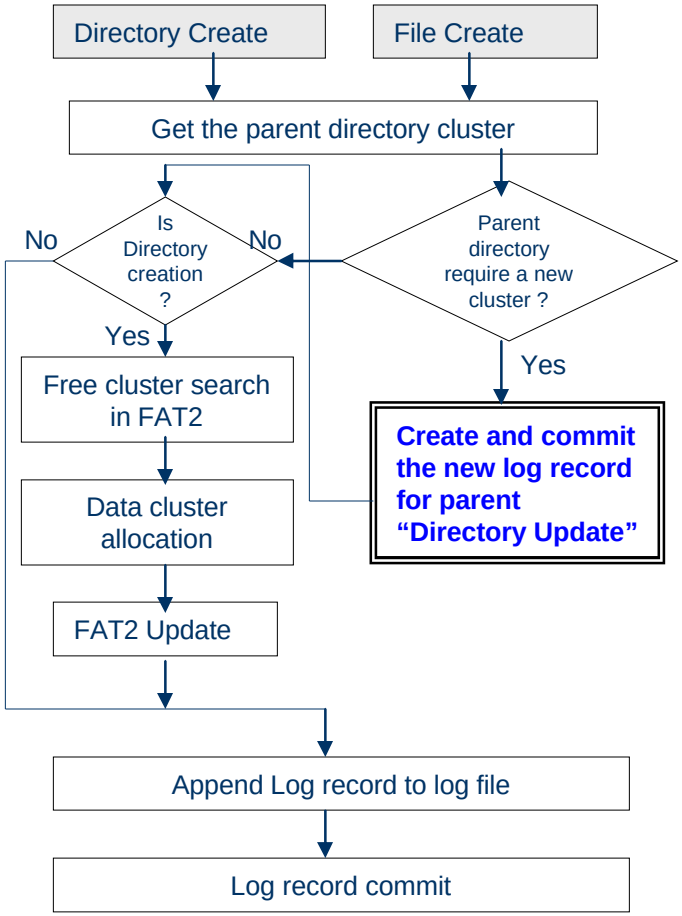
- Typically the log record with file/directory creation or rename operation includes additional log records with Long File Names (LFN).
- Each additional log record can contain a maximum of 16 Unicode characters of LFN.
- The checksum is calculated while committing the log record to file system.
- After the Log record Committed to file system, The first byte of DIR_name field of set as 0xE5 Indicating that log record is committed.
- Indicating the log record as committed by writing 0xE5 as first byte of DIR_name is same as marking the FAT 32 byte deleted during file/directory deletion.

Logging and Committing



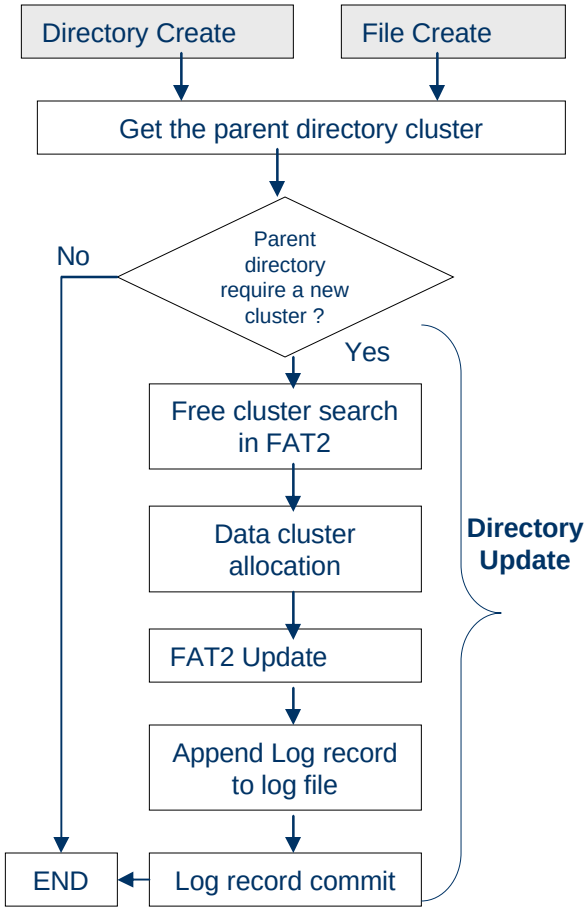
- Logging is performed during execution of following file system operations
  - File/Directory Create,
  - File/Directory Delete,
  - File/Directory Rename
  - File Truncate
- Committing is also done in above operations
- In case of File write operation only the FAT2 is updated.
- Incase of file close (file write is invoked before) logging and commit operations are performed.
- The File/Directory read, file open in read only, getting file system statistics operations need not be logged, because these operations does not update/change the file system meta data.
- If Uncontrolled Power loss happens during “Logging”, then there is no file system Update.
- If Uncontrolled power loss happens after logging, then commit is performed in the immediate reboot.

File/Directory Create



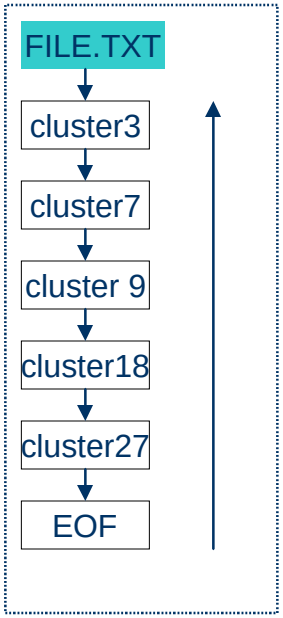
- Logging
  - In case of Directory Create, "." and ".." entries are created in a free cluster and same is marked as allocated in FAT2
  - The Log Record with DIR_log= Create , is appended to Log file/Table.
  - In case of LFN , multiple Log records are appended to Log file
- Commit
  - In case of Directory create, Cluster allocated in FAT2 are copied to FAT1
  - All LFN log records are written to file system as FAT 32 byte directory entries
  - The SFN is calculated and final SFN 32 byte directory entry created
  - The LFN log records are marked as committed
  - The Log record with DIR_log=1 is marked as committed.
- For SFN (Short File Name) Entries, the FAT file system uses following values in DIR_NTRes
  - 0x08 – All 8 characters of name of DOS8.3 name are lower case letters
  - 0x10 – All 3 characters of extension of DOS8.3 name are lower case letters
  - 0x18 – All 8 characters of name and extension of DOS 8.3 are lower case letters.
- During commit operations, the value of field DIR_NTRes is calculated and also the date and time is written to 32 byte directory entry.
- While creating a file/directory, there could be addition of new cluster to parent directory; then a new cluster is allocated in FAT2 and log record with DIR_log = Update, should be included to log file and same should be committed before the file/directory create log record creation and commit.

Directory Update



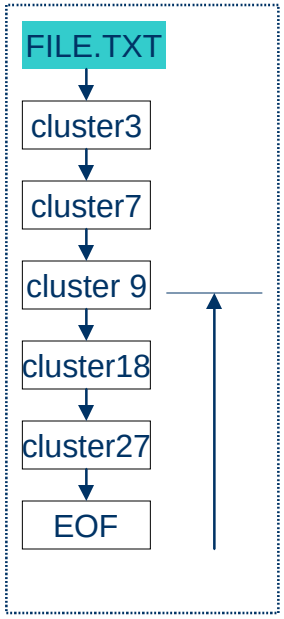
- Might require while creating a new file or directory.
- While creating new file/directory, if there is required number of 32 byte directory entries are NOT free in parent directory (directory in which file/directory needs to be created) then "**Directory Update**" needs to be done.
- Directory update means appending the new cluster to linked list of the data clusters of a directory.
- Logging and Committing of Directory Update should happen before the Logging of new file/directory creation.

File/Directory Delete



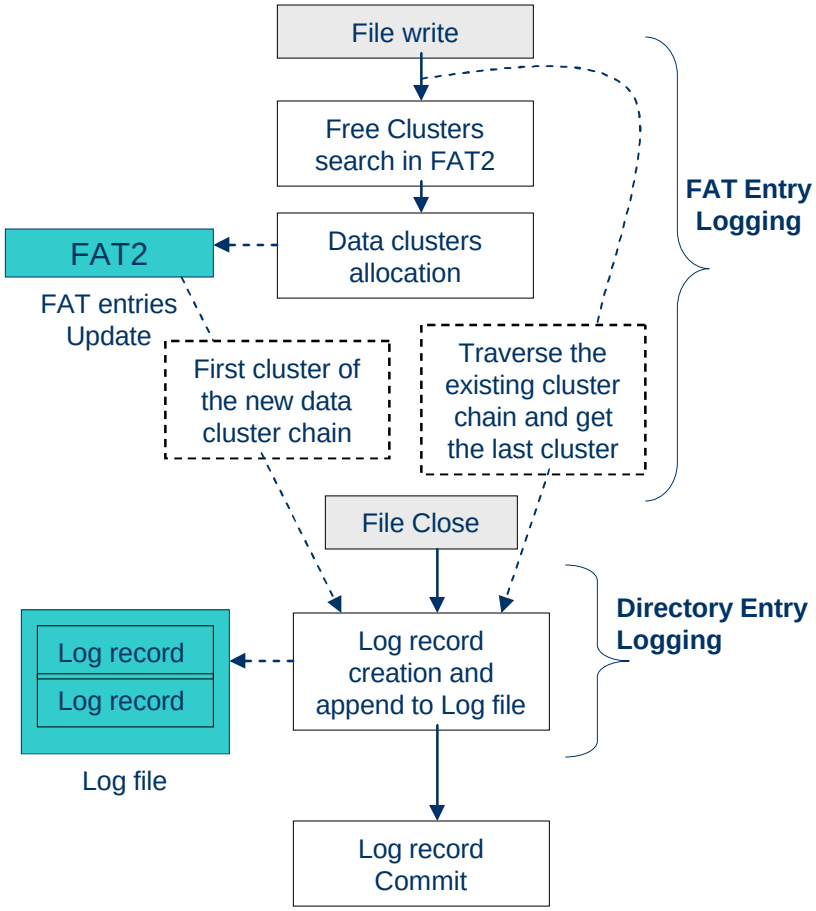
- The Log record with DIR_log = Delete is included to the log file.
- Along with the log record to ensure the safe file delete following algorithm is required
  - Get the first cluster of the data cluster chain of file/directory to be deleted, by reading FAT 32 byte directory entry.
  - Traverse the cluster chain completely and collect all clusters
  - Free the cluster entries in FAT from last cluster to first cluster; instead of from first to last cluster
  - Once all clusters are freed in FAT ; then mark the log record as committed.
- This mechanism ensures that if the power fail occurs while freeing the cluster chain; then in next reboot read the same uncommitted log record and execute the same algorithm.
- This algorithm prevents occurrence of orphan FAT entries in case uncontrolled power loss occurred during file/directory delete
- In this example, the cluster freeing in FAT is done from 27 to 3 in the reverse order; instead of freeing from cluster 3 to 27.

File Truncate



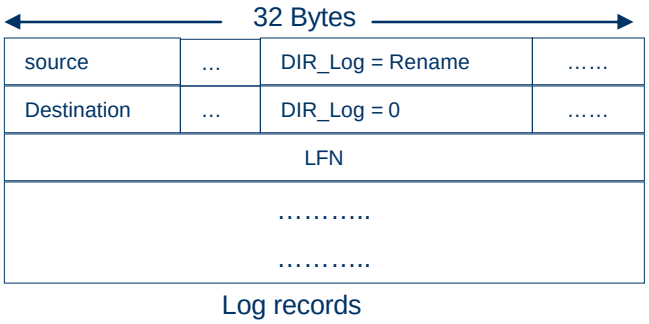
- The Log record with DIR_log = Truncate is included to the log file.
- The DIR_Cluster is the starting cluster to truncate.
- Starting from DIR_Cluster, the file/directory deletion algorithm is followed to delete the FAT entries.
- DIR_cluster is made as the last cluster of the cluster chain.
- In the example, the DIR_cluster is 9 and starting from this cluster the file/directory deletion algorithm is applied to delete the FAT entries.
- A new file size is updated while committing the log record.

File Write and Close



- During file write, only FAT entries are created. This means the newly allocated clusters chain created in FAT2.
- While allocating new clusters, searching for new free cluster always performed in FAT2.
- While closing file, Log record is created with DIR_cluster indicating the starting cluste of new cluster chain; The DIR_LstClusHI/ DIR_FstClusHI and DIR_LstClusLO/ DIR_FstClusLO indicating the last cluster of the file before update.
- While committing the log record,
  - The cluster chain starting with DIR_cluster is copied to FAT1,
  - and same cluster chain is linked to existing cluster chain whose last cluster is specified fields DIR_LstClusHI/ DIR_FstClusHI and DIR_LstClusLO/ DIR_FstClusLO of log record.

File/Directory Rename



Log records

- 1st Log Entry is to delete
- 2nd Log entry is to create
- LFNs exists from 3rd log entries.

- At least two log records are created.
- The first log record will have DIR_log = Rename and the second log record will have DIR_log = 0 indicating the same rename operation.
- The second log record is dependent on first log record.
- If LFN exists in a new file/directory, then second log record contain the additional LFN log records.
- if file/directory is moving (renaming) to different folder/directory, then DIR_Dcluster of first log record is cluster of the source directory and DIR_Dcluster of second log record is the cluster of the destination directory.

References

- FAT Specification: http://msdn.microsoft.com/en-us/windows/hardware/gg463080
- TFAT: http://msdn.microsoft.com/en-us/library/aa915463.aspx
- TFAT patent number 7174420 : http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=/netahtml/PTO/srchnum.htm&r=1&f=G&l=50&s1=7174420.PN.&OS=PN/7174420&RS=PN/7174420
- KFAT : http://www.ksea-ci.org/UKC2005/UKC2005.htm
- RFS : http://www.samsung.com/global/business/semiconductor/products/fusionmemory/Products_FAQs_RFS.html
- RFS power off recovery: http://www.samsung.com/global/business/semiconductor/products/fusionmemory/downloads/RFS_POR_10.pdf
- TFS 4 : http://www.samsung.com/global/business/semiconductor/products/fusionmemory/Products_TFS4_ApplicationNotes.html
- TFS 4 power off recovery: http://www.samsung.com/global/business/semiconductor/products/fusionmemory/downloads/tfs4_v16_power_off_recovery_rev10.pdf

Questions

Queries and Feedback

- keshava_mgowda@ti.com
- keshava.gowda@gmail.com