How does the Windows OS File System Work?

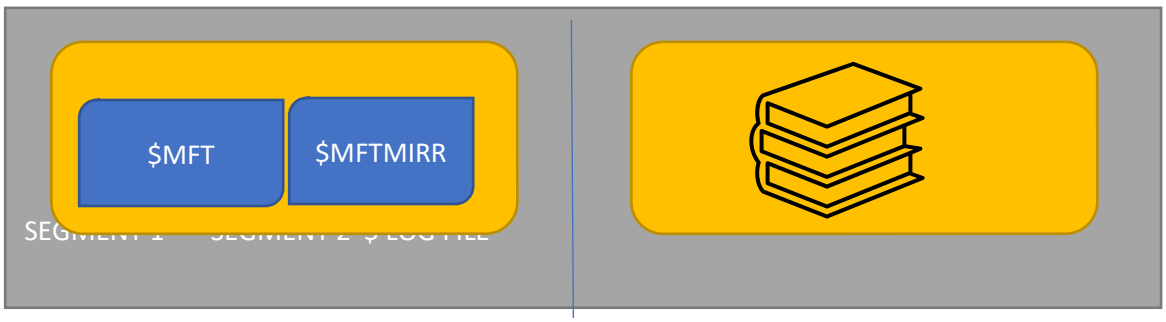
Author:Casey Munga

One of the most import features in Windows File System is journaling. The NTFS uses a journaling system to track the data in an intermediate state before serialization and keeps a record of all the inputs and outputs to its system. The major use of journaling is to ensure that the data, after a system crash or failure, is atomic in its recovery. It is critical that the maximum data recovered is correct and the data is loaded efficiently and with minimum delay and data loss.

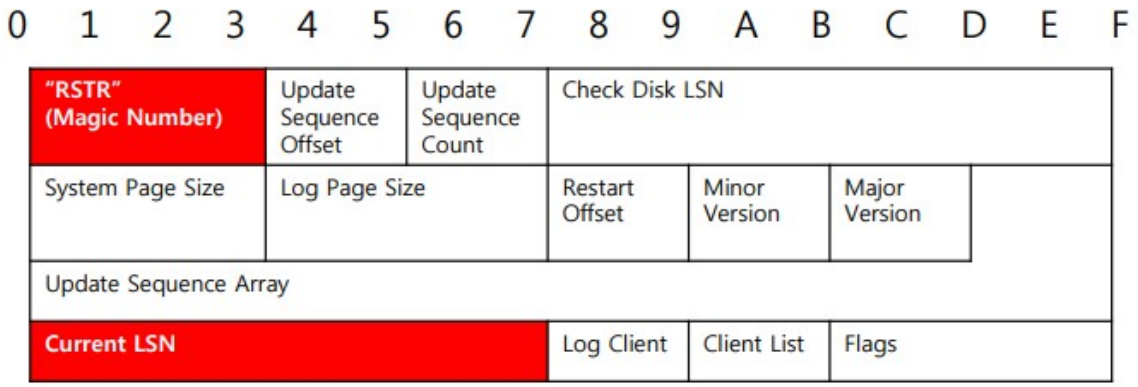
Historically, journaling is named after the concept of journaling in a diary. The Master File Table stores a log of the metadata depicting when files are created, updated, and deleted. For the purposes of describing journaling only the first three records of the Master File Table Record (MFTR) will be referenced. Contained at segment 0 is the Master File Table denoted by the filename $MFT, which keeps a history of each file that exists on the volume. Segment 1, houses file $MFTMIRR File, which is a copy of segment 0. The $LOGFILE is located in segment 2 of the Master Boot Record (MBR) and cannot be accessed by the user. If an attempt is made, the system reveals a “Blue Screen of Death” (BSOD) indicating a system failure.

Microsoft, however, has created Tools that can access the $LOGFILE. The file Nfi.Exe called the NTFS File Sector Information Utility, a support tool from Microsoft is a viewer that allows the data and its structure to be observed.

The structure of the Log File Service is separated into two areas, the restart, and the logging area.

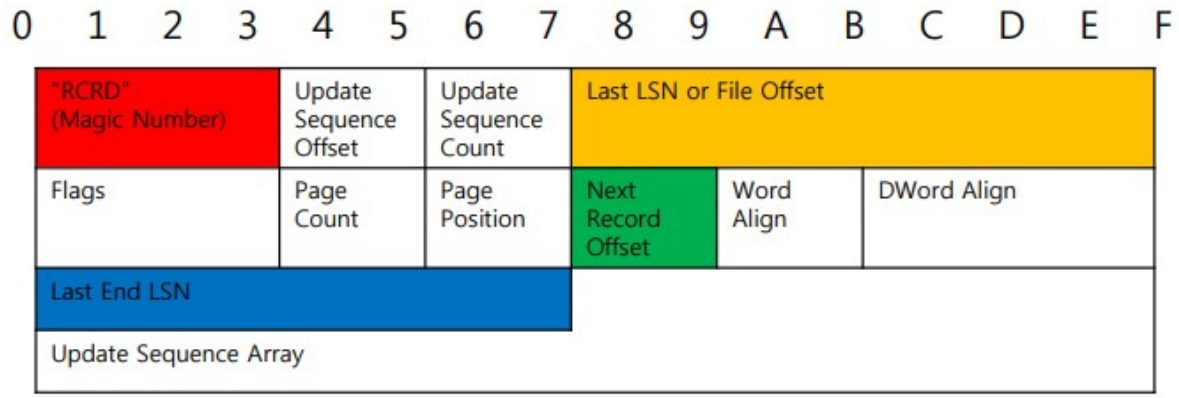


RESTART PAGE



The restart page contains the latest transactional data references that points to the page header to be used as a recovery safe point. A backup copy of the restart page is also stored as insurance that they system would be operable in the case of the restart page's corruption.

PAGE HEADER



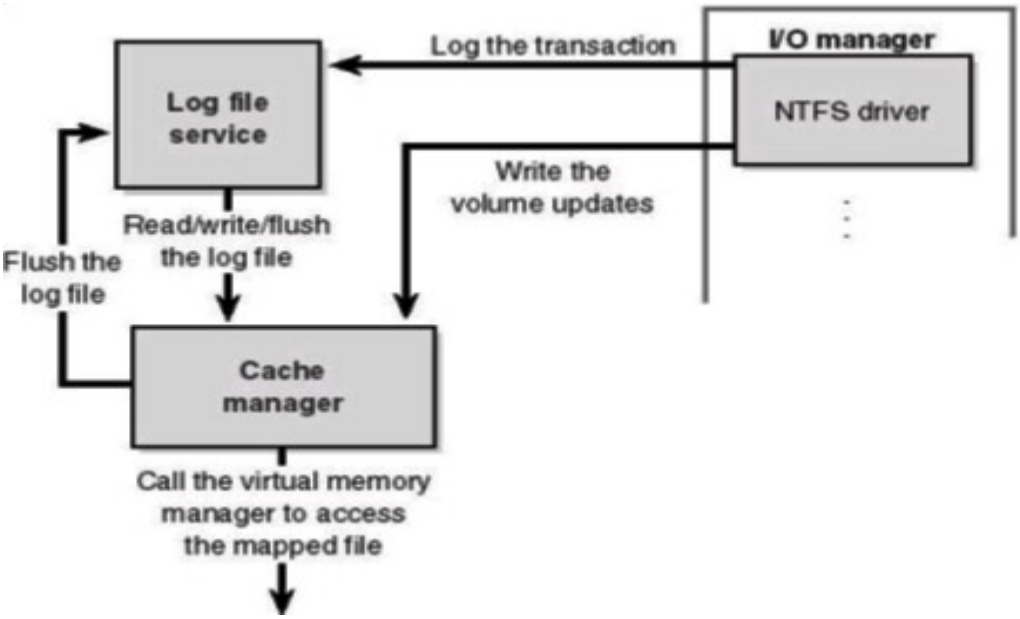
STEPS OF JOURNALING

A buffer records the changes implemented to the file system, which is temporarily recorded in the log known as a write-ahead transaction and generates a 64bit Log Sequence Number (LSN). The LSN Number is accompanied by an offset which points to the LSN Record. Each LSN is sequentially generated in increasing numerical sequence with the newest record having the highest value.

0 1 2 3 4 5 6 7	8 9 A B C D E F
This LSN	Previous LSN
Client Undo LSN	Client Data Length / Client ID
Record Type / Transaction ID	Flags / Alignment or Reserved
Redo OP / Undo OP / Redo Offset / Redo Length	Undo Offset / Undo Length / Target Attribute / LCNs to follows
Record Offset / Attr Offset / MFT Cluster Index / Alignment or Reserved	Target VCN / Alignment or Reserved
Target LCN / Alignment or Reserved	

A copy of that transaction data is duplicated to the restart area $MFTMIRR and returns the log sequence file number.

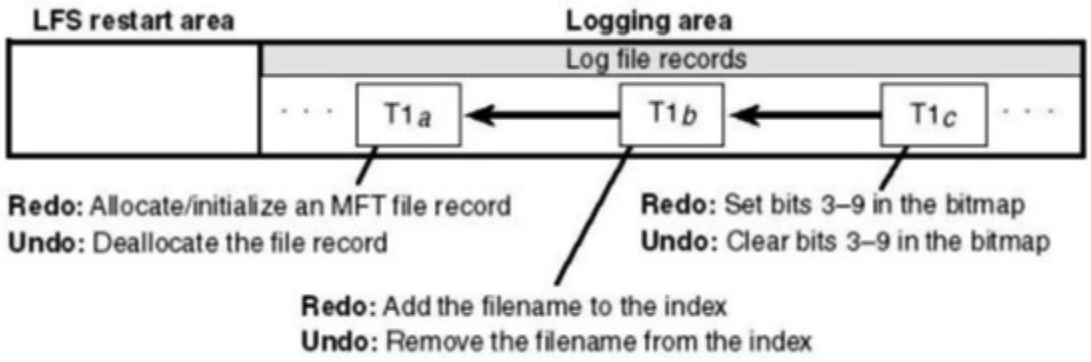
Records are sequentially logged in a circular fashion in the NTFS file system. Each new record is written at the beginning of the head and is released by its tail as the data becomes old. This action prevents a data wrap-around should the log file reach maximum capacity which will contaminate the new head transaction records. In the event that the log file becomes full an exception will be thrown and a roll back transaction will occur.



1. The NTFS driver receives a change to the current file system structure.
2. The NTFS sends the transaction to the Log file service.
3. The Log file service initiates its driver core routines to obtain entry to the $LOGFILE and stores a copy to the restart data area.
4. Old records are set to be flushed from the Log file records
5. Any transactions that modify the file system structure are entered into the cache.
6. The Cache Manager initiates a call to the Log File Service to acquire pages to flush to the disk
7. The LFS enters a transaction into the log that the transaction has been committed.
8. The Log file position is reset to the current LSN.

SYSTEM RECOVERY

The most important purpose of the $LOGFILE is to safeguard the recoverability of the system. To ensure that the state of the recovery will be an almost exact replica of the file system prior to the system crash or error, the log files must have accurately recorded every transaction before they are serialized to the disk. Each step of journaling is codified in the cache. Any updates to the disk volume results in a commit status to be flagged. This action is also recorded in the $LOGFILE signifying that the write or update has been finalized to the secondary storage volume. In the event of a failure, NTFS accesses the $LOGFILE's transactions, each that had not been serialized will be rolled back.



Log file records are of two categories, the redo, and the undo. Redo entries are records that must be reloaded in the aftermath of a non-recoverable system failure and the data has not yet been serialized. The indicated transaction in progress before the failure must be restarted.

The Undo entries indicate transactions that must be rolled back to their pre-crash state during the recovery as they may be incomplete. All bad sectors will be recovered by Cluster Mapping. Clusters are portions of the disk partitions that has been separated into logical entities dependent on the disk capacity. NTFS selects clusters with 64 bit addresses and references it by a table stored in the $Bitmap which resides at segment 6 in the MFT. In addition to the $LOGFILE to ensure that the recovery avoids these bad sectors when the files are allocated, ensuring minimal data loss and system stability.

Glossary:

NTFS:

- **New Technology File System : proprietary software from Microsoft which keeps track of all changes to data at the metadata level.**

BSOD:

- **Blue Screen of Death : Microsoft screen that is displayed to the user to indicate that the system has malfunctioned.**

LSN:

- **Log Sequence Number: the number that identifies the current record that is logged**

References:

31, M. (2018, September 18). Inside NTFS. Retrieved August 02, 2020, from https://www.itprotoday.com/windows-78/inside-ntfs

Batchelor, D., & Satran, M. (2018, May 31). Using the Change Journal Identifier - Win32 apps. Retrieved August 02, 2020, from https://docs.microsoft.com/en-us/windows/win32/fileio/using-the-change-journal-identifier

Cowen, D., & Seyer, M. (2012, May). File System Journal Analysis. Retrieved August 2, 2020, from https://digital-forensics.sans.org/summit-archives/DFIR_Summit/File-System-Journaling-Forensics-Theory-Procedures-and-Analysis-Impacts-David-Cowen-with-Matthew-Seyer.pdf

Greg-Lindsay. (n.d.). Log files - Windows IT Pro - Windows Deployment. Retrieved August 02, 2020, from https://docs.microsoft.com/en-us/windows/deployment/upgrade/log-files

Huculak, M. (2016, October 21). Using Resilient File System (ReFS) on Windows 10. Retrieved August 02, 2020, from https://www.windowscentral.com/how-use-resilient-file-system-refs-windows-10

Msuhanov, ~. (2019, February 17). How the $LogFile works? Retrieved August 02, 2020, from https://dfir.ru/2019/02/16/how-the-logfile-works/

NTFS Master File Table (MFT). (n.d.). Retrieved August 02, 2020, from http://www.ntfs.com/ntfs-mft.htm