Understanding file systems

Save/Share it on:

Presently, the computer market offers a variety of opportunities of storing huge amount of personal or corporate information in digital form. Storage devices include internal and external hard drives, USB flash drives, memory cards of photo/video cameras, complex RAID-systems etc. Actual documents, presentations, pictures, music, video, databases, email messages are stored in a form of files which may be place-consuming.

The following article provides detailed description of how information is stored on a storage device.

Content

- What is a file system?
- Windows file systems
- MacOS file systems
- Linux file systems
- BSD, Solaris, Unix file systems
- Clustered file systems

What is a file system?

Any computer file is stored on some kind of a storage with a given capacity. Actually, each storage is a linear space for reading or both reading and writing digital information. Each byte of information on the storage has its own offset from the storage start (*address*) and is referenced by this *address*. A storage can be presented as a grid with a set of *numbered cells* (each cell is a single byte). Any file saved to the storage gets a number of these cells.

Generally, computer storages use *a pair of sector* and *in-sector offset* to reference any byte of information on the storage. **The sector** is a group of bytes (usually *512 bytes*) that is a minimum addressable unit of the physical storage. *For example*, byte *1040* on a hard disk will be referenced as a sector *#3* and *offset in sector 16 bytes* ([sector]+[sector]+[16 bytes]). This scheme is applied to optimize storage addressing and use a smaller number to reference any portion of information on the storage.

To omit the second part of the address (in-sector offset), files storing begins *in the sector start* and *occupy the whole sectors* (e.g.: 10-byte file occupies the whole sector, 512-byte file also occupies the whole sector, at the same time, 514 byte file occupies two whole sectors).

Each file is stored on *'unused' sectors* and can be read then by a known position and size. However, how do we know what sectors are used or unused? Where are file size and position stored? Where is a file name? These answers are given by **a file system**.

As a whole, *file system* is structured data representation and a set of **metadata** describing the stored data. A file system not only serves for the purposes of the whole storage but is also a part of an isolated storage segment – **disk partition**. Usually, the file system operates *blocks*, not sectors. **File system blocks** are groups of sectors which optimize

storage addressing. Modern file systems generally use block sizes from 1 to 128 sectors (512-65536 bytes). Files are usually stored from the start of a block and take entire blocks.

Immense *write/delete operations* to file system cause file system **fragmentation**. As a result files aren't stored as whole fragments and are divided into fragments. *For example*, a storage is entirely taken by files with size about 4 blocks (e.g. a collection of pictures). User wants to store a file that will take 8 blocks and therefore deletes the first and the last files. By doing this it clears the space on 8 blocks, however, the first segment is near to the storage start, and the second – to the storage end. In this case 8-block file is split into two parts (4 blocks for each part) and takes free space 'holes'. The information about both fragments as parts of a single file is stored to file system.

In addition to user's files, the file system also contains its own *parameters* (such as block size), *file descriptors* (including file size, file location, its fragments etc.), *file names* and *directory hierarchy*. It may also store security information, *extended attributes* and other parameters.

To comply with diverse requirements as storage performance, stability and reliability plenty of file systems are developed to serve certain user purposes.

Back to top

Windows file systems

Microsoft Windows OS uses two major file systems: **FAT**, inherited from old DOS with its later extension **FAT32**, and widely-used **NTFS** file systems. Recently released **ReFS** file system was developed by Microsoft as a new generation file system for Windows 8 Servers.

FAT:
FAT (File Allocation Table) is one of the simplest types of a file system. It consists of a file system *descriptor sector* (boot sector or superblock), *a file system block allocation table* (referred as File Allocation Table) and *plain storage space* to store files and folders. Files on FAT are stored in directories. Each directory is an array of *32-byte records*, each defining file or file extended attributes (e.g. long file name). File record attributes the first block of a file. Any next block can be found through a block allocation table by using it as linked-list.

Block allocation table contains an array of block descriptors. *Zero* value indicates that the block is not used and *non-zero* – relates to the next block of the file or a special value for file end.

The numbers in *FAT12*, *FAT16*, *FAT32* stands for the number if bits used to enumerate file system block. This means that **FAT12** can use up to 4096 different block references, while **FAT16** and **FAT32** can use up to 65536 and 4294967296 accordingly. Actual maximum count of blocks is even less and depends on implementation of *a file system driver*.

FAT12 was used for old **floppy disks**. *FAT16* (or simply FAT) and *FAT32* are widely used for **flash memory cards** and **USB flash sticks**. The system is supported by mobile phones, digital cameras and other portable devices.

FAT or *FAT32* is a file system, used on Windows-compatible external storages or disk partitions with *size under 2GB (for FAT) or 32GB (for FAT32)*. Windows cannot create FAT32 file system over 32GB (however Linux supports FAT32 up to 2TB).

NTFS:

NTFS (New Technology File System) was introduced in Windows NT and is major file system for Windows at present. This is a default file system for disk partitions and the only file system that is supported for disk partitions *over 32GB*. The file system is quite extensible supporting many file properties, including *access control*, *encryption* etc. Each file on NTFS is stored as file descriptor in a *Master File Table* and file content. **Master file table** contains all information about the file: size, allocation, name etc. The first and the last sectors of the file system contain *file system settings* (boot record or **superblock**). This file system uses *48* and *64 bit* values to reference files, thus supporting quite large disk storages.

ReFS:
ReFS (Resilient File System) is the latest development of Microsoft presently available for Windows 8 Servers. File system architecture absolutely differs from other Windows file systems and is mainly organized in a form of **B+-tree**. *ReFS* has high tolerance to failures due to new features included into the system. And, namely, **Copy-on-Write** (CoW): no metadata is modified without being copied; data is written not over the existing data, but into a new disk space. With any file modifications a new copy of metadata is created into free storage space, and then the system creates a link from older metadata to the newer ones. As a result, a system stores significant quantity of older backups in different places providing easy file recovery unless this storage space is overwritten.

For information about data recovery from these file systems please visit Deleted files: chances for recovery page.

Back to top

MacOS file systems

Apple Mac OS operating system applies *HFS+* file system, an extension to their own HFS file system that was used on old Macintosh computers.

HFS+ file system is operatedby *Apple* desktop products, including Mac computers, **iPhone**, **iPod**, as well as Apple X Server products. Advanced server products also use Apple Xsan file system, *clustered file system* derived from StorNext or CentraVision file systems.

This file system stores files and folders as well as *Finder* information about directories view, window positions etc.

For information about data recovery from these file systems please visit Deleted files: chances for recovery page.

Back to top

Linux file systems

Open-source Linux OS aims at implementing, testing and using different concepts of file systems. The most popular Linux file systems nowadays are:

- **Ext2, Ext3, Ext4** - *'native'* Linux file system. This file system falls under active developments and improvements. **Ext3** file system is just an extension to **Ext2** which uses transactional file write operations with a **journal**. **Ext4** is a further development of Ext3, extended with support of optimized file allocation information (extents) and extended file attributes. This file system is frequently used as a *'root'* file system for most Linux installations.

- **ReiserFS** - alternative Linux file system is created *to store huge amount of small files*. It has good capability of files search and enables compact files allocation by storing file tails or small files along with metadata in order not to use large file system blocks for the same purpose.
- **XFS** - file system derived from SGI company and was initially used for company's IRIX servers. Now XFS specifications are implemented in Linux. XFS file system has great performance and is widely used to store files.
- **JFS** - file system is developed by IBM for the company's powerful computing systems. **JFS1** usually stands for **JFS**, **JFS2** is the second release. Currently this file system is *open-source* and implemented in most modern Linux versions.

The concept of **'hard links'** used in this kind of OS is a common feature of Linux file systems of not regarding the file name as file attribute and rather defined as an alias for a file in certain directory. File object can be *linked from many locations*, even multiply from the same directory under different names. This may present serious and even insurmountable difficulties for recovery of file names after file deletion or file system damage.

For information about data recovery from these file systems please visit Deleted files: chances for recovery page.

Back to top

BSD, Solaris, Unix file systems

The most common file system for these OS is UFS (Unix File System) also often referred to as *FFS* (Fast File System – fast compared to a previous file system used for Unix). **UFS** is a source of ideas for many other file system implementations.

Currently UFS (in different editions) is supported by all Unix-family OS and is major file system of BSD OS and Sun Solaris OS. Modern computer technologies tend to implement replacements for UFS in different OSs (**ZFS** for Solaris, **JFS** and derived file systems for Unix etc.).

For information about data recovery from these file systems please visit Deleted files: chances for recovery page.

Back to top

Clustered file systems

Clustered file systems are used in computer cluster systems. These file systems provide support of distributed storage.

Such distributed file systems include:

- **ZFS** - Sun company *'Zettabyte File System'* - a new file system developed for distrubuted storages of Sun Solaris OS.
- **Apple Xsan** - the Apple company evolution of CentraVision and later StorNext file systems.
- **VMFS** - *'Virtual Machine File System'* developed by VMware company for its VMware ESX Server.
- **GFS** - Rad Hat Linux *'Global File System'*.

- **JFS1** - original (legacy) design of *IBM JFS* file system used in older AIX storage systems.

Common property of these file systems includes distributed storages support, extensibility and modularity.