International Conference on Intelligent Control and Computer Application (ICCA 2016)

Research on Allocator Strategy of FAT32 File System Based on Linux & Windows

Li Qiyang, Zhang Quanxin, Tan Yu-an, Li Yuanzhang*, Zheng Jun

Beijing Engineering Research Center of Massive Language Information Processing and Cloud Computing Application,
School of Computer Science and Technology, Beijing Institute of Technology
Beijing 100081, China
*Corresponding author: popular@bit.edu.cn

***Abstract*—As FAT32 file system implemented on Windows is not open source, this paper tries to analyze its allocator strategy by analyzing the structure and principle of DBR, FSINFO sectors on FAT32, as well as the source code of FAT32 file system implemented on Linux. Then using these theories together with hypotheses to do massive experiments of creating, copying and deleting files on devices with different format type to help analyze it in depth. Declare the differences by comparing it with three common allocator strategies, finally help researchers and users to understand FAT32 file system better and protect their data better.**

***Keywords*—FAT32 file system; allocator strategy; DBR; FSINFO**

I. Introduction

Nowadays, with the decrease of the cost of flash chips and the popularity of smart phones, as well as other storage devices, such as tablet computer, digital camera, USB flash disk and SD card, have filled every part of human being's life. More and more data is generated during people's daily life, so the security of the data becomes more and more important [1]. But it's such a wide variety of different storage devices and the use scenes tend to be diverse and complex, all these make the storage situation on storage devices become more complex and mysterious [2]. If user doesn't know the storage situation on their own devices or doesn't aware it, there may have potential safety problems which can cause great losses [3]. Malicious program or data may be hidden in user's data, which is a great threaten.

II. Background and Related Work

A. Three common Allocator algorithms

There are three common types of allocator algorithms used in file system allocator strategy: first available, next available and best fit [4]. It has been tested that, the Windows 98 and XP operating systems appear to use a next available algorithm for FAT file systems.

The FAT file system derives its name from the file allocation table structure (FAT), which is essentially an array of annotated implicit references to the clusters (groups of consecutive disk sectors) with the file system. For each file, the FAT contains a linked list of clusters – these lists are called FAT chains.

A first available strategy searches for an available data unit starting with the first data unit in the file system. After a data unit has been allocated using the first available strategy and a second data unit is needed, the search starts again at the beginning of the file system. This type of strategy can easily produce fragmented files because the file is not allocated as a whole.

A similar strategy is next available, which starts its search with the data unit that was most recently allocated instead of at the beginning. Next available strategy won't reuse the space that freed by deleting files before, it will go back to the beginning cluster after reaching the last cluster on the device.

The first available strategy and next available strategy all fill in the data when find the first free cluster, and ignore the continuity of file data storage. The best fit strategy searches for consecutive data units that fit the needed amount of data [5].

B. Analyze on DBR & FSINFO sector in FAT32 file system

In FAT32 file system, there're two sectors to record information about file system [6]. DBR sector is the first sector in each partition, it is used to record general and static information, such as "bytes per sector", "sectors per cluster", "cluster number of root directory" and so on, which stay stable after being formatted by operating system. There's an attribute named "BPB_FSInfo" points to the position of FSINFO sector.

FSINFO sector stores the current information of file system, providing to the operating system as a reference [7]. "Free cluster number" and "Next available cluster" are recorded in it, which change with file read and write operation. Among them one attribute called "FSI_Nxt_Free" indicates the next available cluster, which helps keep the allocation consistent and completed [8]. It locates at 0x1EC bytes offset in FSINFO sector and takes 4 bytes space.

C. Related works

There's few research on FAT file system allocator strategy implemented on Windows, as it is not open source. Wicher Minnaard who comes from Netherlands Forensic Institute has done a deep analyze on FAT file system allocator strategy implemented on Linux [9]. He also combines it with the file

This work is supported by the National Natural Science Foundation of China (61272511, 61370063); National High Technology Research and Development Program of China (863 Program, 2013AA01A212); Beijing Higher Education Young Elite Teacher Project, YETP1178;SRF FOR ROCS, SEM.

creation order to get more insights, but finally he didn’t drill down to the exact position that the data write on devices [10]. Kesshava Munegowda, G.T. Raju and Veera Manikandan Raju also have done some investigation on allocator strategy, but they mainly focused on ExFAT file system and did some comparison with FAT32 file system [11]. In the paper, they mentioned about the optimization progress of searching for available clusters and the reading algorithm of continuous clusters. Then use black box tests results and public theories, find out a new theory and proved it with experiments [12].

At the present stage, research on allocator strategy is constrained on three common uses strategies and not deep enough. When using on complex scenes, many important details are still not so specific and clear. This paper focus on allocator strategy and storage location when deleting files and creating new files on devices, then find out the real strategy that Windows uses.

III. ALLOCATOR STRATEGY OF FAT32 FILE SYSTEM

FAT32 file system is compatible with many operating systems, Windows and Linux are the two major ones. This paper tries to analyze allocator strategy on Linux and use it as reference, then explore the allocator strategy used on Windows with a series of experiments.

A. Allocator strategy of FAT32 file system used on Linux

In Linux source code, there’s a file (/fs/fat/fatten.c) implements the main functions and operations used in FAT file system. Among them, fat_alloc_clusters() function describes the allocator strategy that operating system uses, in which the building blocks of a next available allocator are recognizable. Struct “msdos_sb_info” stores all the attributes about file system, “prev_free” is one of them, which points to the next available cluster [13].

The main flow is shown in Fig.1, program begins to traverse FAT chain to find the available cluster from “prev_free+1”. While traversing the FAT in this manner, the allocation state of entries is evaluated. An entry marked as free will become allocated and attribute “prev_free” will be set as new entry. Searches continue until the requested amount of clusters is satisfied. If a search reaches the end of the volume, it is wrapped around, the search will then be restarted at the beginning of the data area of the volume (cluster 2).

After analyzing the source code of FAT file system on Linux, we find that the attribute “prev_free” is never modified outside of the allocator function except being initialized at the beginning. Which means this attribute will thus only ever increase until it reaches to the maximum and so will the starting point of a search for available clusters.

In conclusion, the allocator strategy of FAT32 file system used on Linux is the next available strategy, which always starts to search for available cluster from position that the hint indicates, then wraps around when reaching the end of the volume.

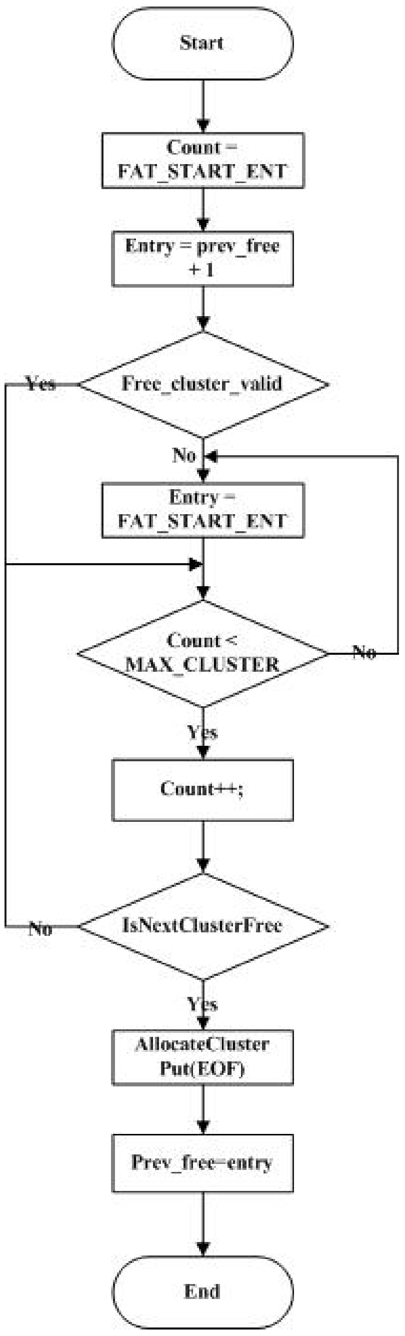


Fig. 1. Work flow of FAT32 file system allocator strategy on Linux

B. Allocator strategy of FAT32 file system used on Windows

Although source code of FAT32 file system implemented on Windows is invisible, we can explore the allocator strategy by referring to the method that implemented on Linux and together with analysis during experiments. Finally, we find that allocator strategy used on Windows is neither any one of three common strategies, but a more complex one.

There’re two main differences between the strategy implemented on Linux and Windows:

1) Computational method of next available cluster is different

On Linux the next available cluster is stored in the field of "FSI_Nxt_Free" in FSINFO sector, every time operating system can access it directly and start a search. But on windows, except the minimum physical storage unit "sector" and the minimum space unit that a file takes "cluster", there exists another storage unit, like "block". Every "block" consists of 65536 clusters (shows in Fig.2), the value of "FSI_Nxt_Free" in FSINFO sector varies between 0 to the maximum number of cluster that a "block" can hold (65535). The real cluster number of the next available cluster is calculated as (1):

$$\text{Next available cluster No.} = 65526 * \text{current "block" No.} + \text{FSI\_Nxt\_Free}; \quad (1)$$

What's more, the "FSI_Nxt_Free" attribute is not always increasing continuously, when deleting files and available space released, the value may be effected. While all the files copied are in a single "block", if delete one of them, the value of "FSI_Nxt_Free" value will be updated to the first cluster of the deleted file immediately; if delete one file which crosses the boundary of the "block", the value of "FSI_Nxt_Free" will be updated to 2. Actually the next available cluster is the first cluster of current "block".

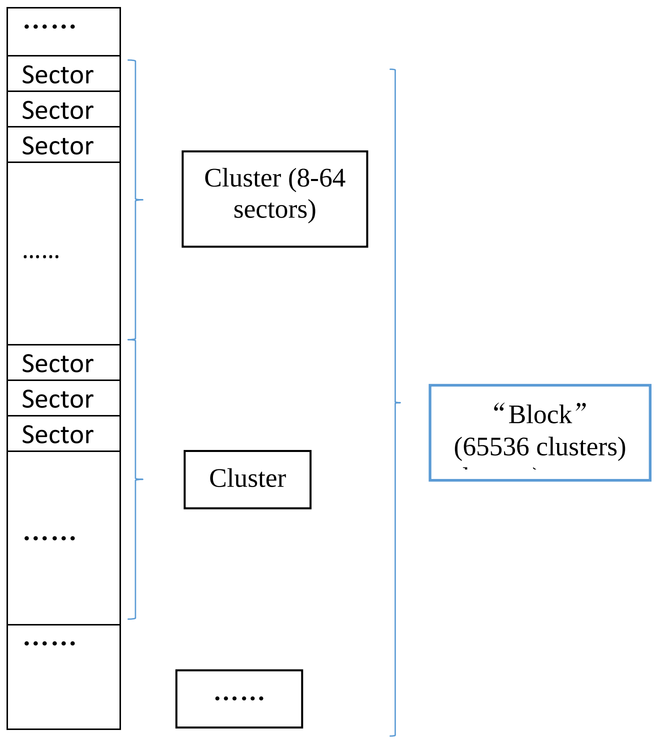


Fig. 2. Organizational form of "Block" of FAT32 file system on Windows

2) Allocator strategy of FAT32 file system implemented on Windows tries to maintain the continuity of the file stored on devices

When deleting a file in the "block" or crossing the "block" boundary and copy another new file to the device. If the size of the new file is larger than the space that released, file system will compare the released space to the space that remains in current "block".

An example shows in Fig.3, storage units in gray have been allocated and filled with data, storage units in white are available. Delete some files and release continuous space ①, copy a new file which size is larger than the size released. If the size of space ① is larger than the size of remaining space ②, operating system will write the new file data from the beginning of space ①, the rest write to the following part; but if the size of space ① is smaller than the size that remains, then the new file data will be written from the beginning of space ② until all the data has been written completely. So the storage units are continuous.

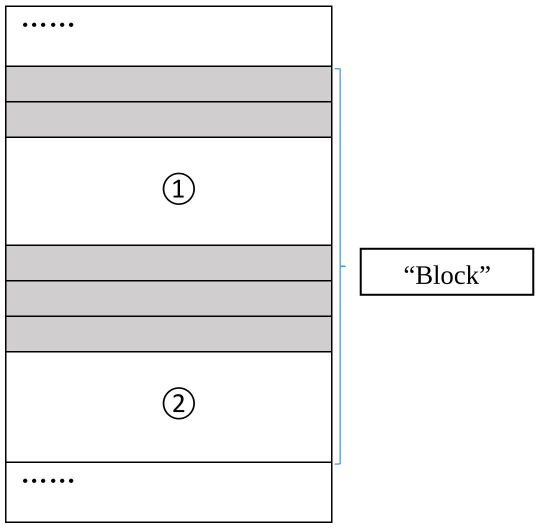


Fig. 3. Decisions when allocate new storage unit in "Block"

IV. KEY POINT ANALYSIS OF ALLOCATOR STRATEGIES

To verify the rationality of conclusion we made before, we have done tons of experiments. Using SD cards with different capacities and different brands, then format into different cluster sizes and start the experiment. Information about all the storage devices used are listed in TABLE I.

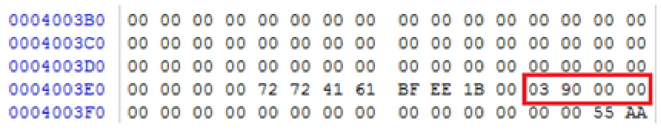
TABLE I. INFORMATION ABOUT ALL THE SD CARD USED

Brand	Type	Cluster	Capacity	Speed Class	Max Speed
Kingston	SD card	8KB	16GB	Class 4	8MB/s
SanDisk	SD card	16KB	32GB	Class 10	30MB/s
Kingston	SD card	32KB	32GB	Class 10	30MB/s

A. "Block" size verification and "FSI_Nxt_Free" update situation

To prove the existence of the "Block" in FAT32 file system implemented on Windows and find out the real size of it, we use two SD cards with the same capacity, then format them into different cluster sizes. One is 16KB per cluster (Card 1), the other is 32KB per cluster (Card 2). As one "block" consists of 65536 clusters, so the size of one "block" of Card 1 is 1GB, and 2GB of Card 2. Do the following experiment:

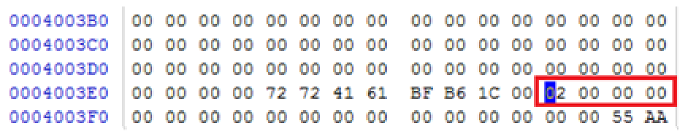
Fill in three files to Card 1, the size of the files is 300 MB, 800MB and 500MB. After finish copying files, use Winhex to check the value of "FSI_Nxt_Free" attribute in FSINFO sector, which changes to 0x9003(36867). Look for the last cluster of 500MB file, which is No.102402. So if we use the formula summarized before, the next available cluster number is: 65536 * current "block" No. + FSI_Nxt_Free = 65536 * 1 + 36867 = 102403. Which is exactly the next cluster after 500MB file, the result of the experiment matches the conclusion we made before.



```
0004003B0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0004003C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0004003D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0004003E0  00 00 00 00 72 72 41 61  BF EE 1B 00 03 90 00 00
0004003F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 55 AA
```

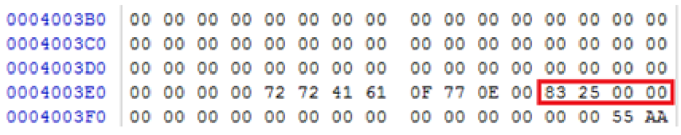
Fig. 4. Value of FSI_Nxt_Free after copying file to Card 1

After that, format Card 1 and Card 2 again and copy the three files to both. Then for Card 1, the 800MB file crosses the boundary of "blocks"; but for Card 2, all the files are still in one "block". If we delete this file, the value of FSI_Nxt_Free attribute in Card 1 will be updated to 2, but in Card 2 it will be updated to the first cluster No. of the deleted file, as it has not reached the end of one "block".



```
0004003B0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0004003C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0004003D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0004003E0  00 00 00 00 72 72 41 61  BF B6 1C 00 02 00 00 00
0004003F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 55 AA
```

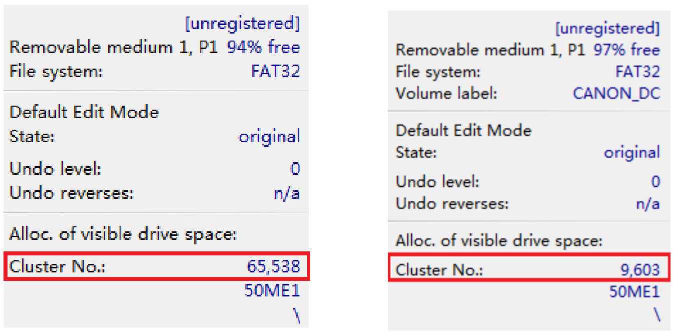
Fig. 5. Value of FSI_Nxt_Free after deleting 800MB file in Card 1



```
0004003B0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0004003C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0004003D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0004003E0  00 00 00 00 72 72 41 61  0F 77 0E 00 83 25 00 00
0004003F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 55 AA
```

Fig. 6. Value of FSI_Nxt_Free after deleting 800MB file in Card 2

Then copy a new file, which size is 50MB, use the formula summarized before, the next available cluster number of Card 1 is: 65536 * 1 +2 = 65538; the next available cluster number of Card 2 is: 65536 * 0 + 9603(0x2583) = 9603. After finish copying the new file, use Winhex to see the position of 50MB file, which are shown in Fig.8. The results are exactly the same with the ones calculated before.



```
                         [unregistered]
Removable medium 1, P1        94% free
File system:                     FAT32

Default Edit Mode
State:                        original

Undo level:                          0
Undo reverses:                     n/a

Alloc. of visible drive space:

Cluster No.:                    65,538
                                 50ME1
                                     \
```

```
                         [unregistered]
Removable medium 1, P1        97% free
File system:                     FAT32
Volume label:                 CANON_DC

Default Edit Mode
State:                        original

Undo level:                          0
Undo reverses:                     n/a

Alloc. of visible drive space:

Cluster No.:                     9,603
                                 50ME1
                                     \
```

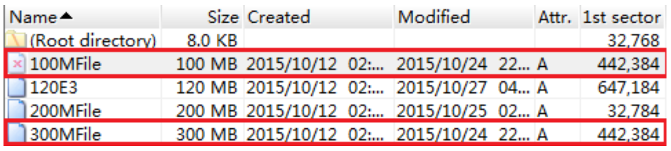
Fig. 7.Firt cluster of 50MB file in Card 1 and Card 2

Format Card 2 again with 32KB per cluster, then copy 4 files in it, which sizes are 300MB,800MB,500MB,700MB, then 700MB file is the one that crosses the boundary of "blocks". Delete this file, we get similar result as that of Card 1 before, the value of FSI_Nxt_Free is updated to 2, then copy 50MB new file, the first cluster is 65538.

In conclusion, there indeed exists a logical organization of clusters in FAT32 file system implemented on Windows, which is similar to "block". It consists of 65536 clusters, and determines the value of "FSI_Nxt_Free" attribute and the allocator strategy of new clusters.

B. *Keep the continuity of file storage verification*

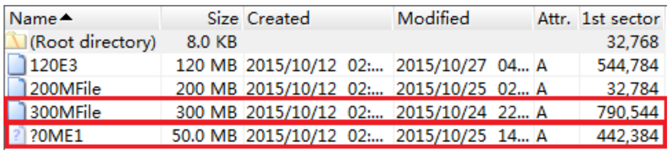
Format the 16GB Card 3 into 8KB per cluster, then the size of one "block" is 65536 * 8KB = 512MB. Copy three files to Card 3, which sizes are 200MB, 100MB and 120MB, so the space left in the first "block" is 92MB. If we delete the 100MB file, and copy a lager file which is 300MB. As the space released by the deleted file is larger than that left in "block", so the new file will be written to the start position of original 100MB file, the rest is written to the following space, the storage of new file is not continuous, the result is shown in Fig.10.



Name▲	Size	Created	Modified	Attr.	1st sector
(Root directory)	8.0 KB				32,768
100MFile	100 MB	2015/10/12 02:...	2015/10/24 22...	A	442,384
120E3	120 MB	2015/10/12 02:...	2015/10/27 04...	A	647,184
200MFile	200 MB	2015/10/12 02:...	2015/10/25 02...	A	32,784
300MFile	300 MB	2015/10/12 02:...	2015/10/24 22...	A	442,384

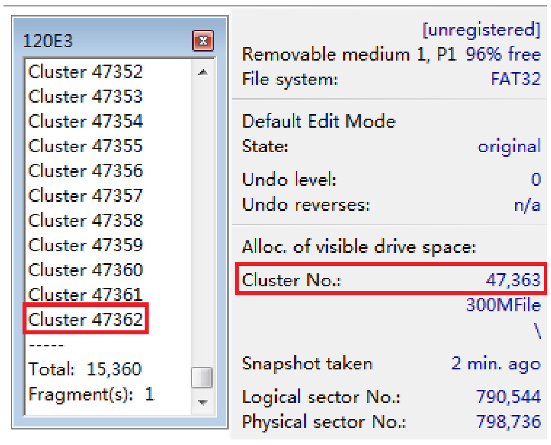
Fig. 8. First cluster No. of 300MB file which copied after deleting file

Format Card 3 again, copy three files which sizes are 200MB, 50MB and 120MB, so the space left in the "block" is 142MB. If we delete the 50MB file, and copy a larger file which is 300MB. As the space released by the deleted file is smaller than that left in "block", so the new file will be written from the first cluster of the left space (Just behind the 120MB file). The storage of new file is continuous, the results are shown in Fig.11 and Fig.12.



Name▲	Size	Created	Modified	Attr.	1st sector
(Root directory)	8.0 KB				32,768
120E3	120 MB	2015/10/12 02:...	2015/10/27 04...	A	544,784
200MFile	200 MB	2015/10/12 02:...	2015/10/25 02...	A	32,784
300MFile	300 MB	2015/10/12 02:...	2015/10/24 22...	A	790,544
?0ME1	50.0 MB	2015/10/12 02:...	2015/10/25 14...	A	442,384

Fig. 9. First sector of new 300MB file is not the first sector of deleted file


120E3

```
Cluster 47352
Cluster 47353
Cluster 47354
Cluster 47355
Cluster 47356
Cluster 47357
Cluster 47358
Cluster 47359
Cluster 47360
Cluster 47361
Cluster 47362
-----
Total:  15,360
Fragment(s):  1
```

```
                                [unregistered]
Removable medium 1, P1              96% free
File system:                           FAT32

Default Edit Mode
State:                              original
Undo level:                                0
Undo reverses:                           n/a

Alloc. of visible drive space:
Cluster No.:                          47,363
                                    300MFile
                                           \
Snapshot taken                    2 min. ago
Logical sector No.:                  790,544
Physical sector No.:                 798,736
```

Fig. 10. First cluster No. of 300MB file which copied after deleting file

In conclusion, when allocate new clusters from FAT32 file system implemented on Windows, the system will consider the storage situation on the device, as well as the size of the file itself, and try to allocate a sequence of continuous space to help keep the continuity of file storage. The allocator strategy is similar to "best fit" strategy, but more complex.

V. CONCLUSIONs

This paper analyzed the allocator strategy used in FAT32 file system implemented on Windows, by analyzing the DBR sector, FSINFO sector in FAT32 file system, as well as referring to FAT32 source code implemented on Linux. Then together with tons of experiments and finally get the conclusion. This makes users and researchers get to know FAT32 better and deeper and also provides a new vision on monitoring data security status, then pushes the progress on refining the information security strategy and building up a more secure file system and storage environment.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (61272511, 61370063); National High Technology Research and Development Program of China (863 Program,2013AA01A212); Beijing Higher Education Young Elite Teacher Project, YETP1178; SRF for ROCS, SEM.

References

[1] Haiyang Fan,Lipeng Wang, Xiaopeng An, Jianbin Gong, Xingfei Shang. Research on recovering data on FAT32 file system [J]. Technology Information,2013,36:55-57.

[2] Brian Carrier. File System Forensic Analysis[M]. Addison Wesley Professional, 2005:12-14

[3] Bugen Huang. Analyzing electronic trail on storage media by file operating for FAT file system[J]. Computer Engineering and Applications,2007,07:233-235.

[4] Microsoft, FAT32 File System Specification, FAT: General Overview of On-Disk Format (2000)

[5] Keshava Munegowda, Veera Manikandan Raju, Madan Srinivas, Rohit Joshi, "FAT file in Reserved Cluster with Ready Entry State", US patent : 8452734, granted on April 19, 2013.

[6] Chen Chao, Jin Huiyun. Study on the Recovery Technology of DOS Boot Recorder on FAT32 File System [J]. NetInfo Security,2011,05:81-83.

[7] Zhang, J.: Research of Embedded FAT file system. In: IEEE International Conference on Uncertainty Reasoning and Knowledge Engineering (2011)

[8] Munegowda, K., Raju, G.T., Raju, V.M.: Performance and Space Optimization techniques for FAT File system for embedded storage devices. In: International Conference on Data Engineering and Communications, ICDECS (December 2011)

[9] SD Specifications Part 1: Physical Layer Simplified Specification version 4.10, SD card Association, January 22, 2013

[10] Wicher Minnaard. The Linux FAT32 allocator and file creation order reconstruction[J]. Digital Investigation,2014,113:.

[11] Munegowda, K., Venkatraman, S., Raju, G.T.: The Extend FAT file system: Differentiating with FAT32 file system. In: Linux Conference, Prague, Czech Republic, Europe (October 2011)

[12] Keshava Munegowda, Venkatraman S, Dr. G T Raju, "The Extended FAT file system: Differentiating with FAT32 file system", Linux Conference, Prague, Czech Republic, Europe, October 2011.

[13] Tweedie, S.C.: Journaling the Linux ext2fs File system. In: Proceedings of the 4th Annual LinuxExpo, Durham, NC (May 1998) (retrieved June 23, 2007)