File System Report

Candidate number: 118440

Date: 3rd March, 2015

1. Introduction

In this report, a brief introduction on a basic FAT (File Allocation Table) file system will be included. Its features shall be explained, followed by the tests it gone through and how it could be improved for future development.

2. Design

In table 2, a description and explanation on the main features of the file system could be found.

Table 2 – Features of the FAT file system

Features	Details
Create New file	Files are stored in blocks and its index is stored in a linked-list way in the FAT (File Allocation Table) inside the metadata. To create a new file, the system would find a free block tin FAT and link the other free blocks if needed. The first index of the file is output for retrieving the file. Files can be read by file names as long as it is under the same memory space.
Read File	To read a file using the starting index, the system first read the first block of information from the disk, followed by continuously searching for linked blocks until it reaches the end-of-file flag in FAT.
Delete File	Deletion of a file only change the flag in the FAT to free a block. This avoids redundant seeking in disk.
Free Space Management	Free space management is implemented to minimize external fragmentation. FAT is used to keep track of free spaces in the disk. Since files are not necessarily allocated in a contiguous pattern, free spaces shall be used in a more practical way. Ideally, every blocks shall be occupied until the disk is full.
Caching	The FAT and other essential metadata are cached into memory for convenient access for the CPU. While the file system is being told to shut down, all the metadata will be write to disk. This enables both sequential and random access to perform well with cached FAT.
Warning for unsafe operations	When the user try to create, read or delete a file, the system shall check if the file exist or if any free space is available. If an unsafe operation is detected, a corresponding exception will be thrown using exceptions classes. This avoids run time error and protects data integrity.

3. Challenge

There are several challenges appeared during the coding stage.

Not knowing the file size is an issue. Since information about the number of blocks won't ensure that the data read from the disk is accurate. When a file is smaller than the block size, not knowing the file size may cause redundant disk reading operation or even reading inaccurate data. A solution to this is to store the file size of each file in the metadata.

Another issue of FAT system is the problem of internal fragmentation. Space are wasted in the disk of most space inside a block is empty. A solution is came up which is to set a smaller block size to maximize available storage.

4. Testing

In order to have a brief analysis on how well the file system works, the following experiments are performed and the results are concluded in table 4:

Table 4 – Experimental result on the performance of the file system

Testing Criteria	Method	Result
Efficiency	Try writing varies length of file and see how the space inside the disk are used.	For a 128bytes disk: Worse case – every file is 0 byte. Used space 32bytes; wasted space 96bytes. Best case – every file equals to the block size. Used space 128bytes; wasted space 0byte.
Limits	Read varied lengths of files and find out the limit	For a 128bytes hard disk - the minimum file size is 0 and maximum is 96 bytes.
Concurrency	Run multiple thread to run the new File method and look the results	Error occurs when two thread trying to write into disk at the same time.
Speed	Time each method and see the performance on individual methods in different disk size.	Formatting a 128bytes disk takes 34ms. Formatting a 1024bytes disk takes 308ms. Writing a 10byte file takes 15ms. Reading a 10byte file takes 21ms. Deleting a 10byte file takes 0ms. Shutting down a 128byte file system takes 41ms.
Reliability	Crash the system during read file operations.	The FAT and metadata in memory is not saved into disk and cannot be recovered.

5. Future Improvements

During the testing stage, several problem of the file system is detected and will be suggested to improve based on the following:

Table 4 – Improvements on future development

Improvements	Problems	Suggested Solutions
Defragmentation	Since disk space allocation in the FAT system is not contiguous, free blocks are scattered randomly across the whole disk. Frequently seeking in the disk when read or write greatly slow down the performance.	Re-arrangement of the blocks occasionally enable files to store in a more contiguous way.
Enhance Reliability	Since FAT is kept inside the memory for quicker access until the signal shutdown method is called, there is potential danger of data lost when encountering a system crash.	Replicate FAT and keep copies on disk.
Ability to expand a file	A file cannot be modified after creation.	Create a new method that enables a file to expend

6. Conclusion

While this FAT file system provide basic functionalities, it still have potentials to be developed into a more sophisticated and useful file system. In the future, it should be aimed to enhance the overall performance and introduce more structural and functional improvements.