

IT308

OPERATING SYSTEM

PROJECT REPORT

File Management System

EKTA BHORANIYA (ID: 201501402)
SAVAN MAKADIYA (ID: 201501418)

Index

1. Introduction	3
1.1 Project Overview	3
1.2 Project Purpose	4
2. Project Description	4
2.1 File Detail Table	5
2.2 File Allocation Table	5
2.3 Features and its commands	6
3. Optimization	8
4. Future Scope	8
5. References	8

1. Introduction

Files are managed by operating system. The operating system uses set of methods and data structures to keep track of files on a disk or partition. Thus the part of operating system dealing with files is called the file system. It is the way files are created, organized and stored on the disk. Major functions of file system includes tracking allocated and free space on disk, maintaining directories and file names and tracking where each file is physically stored on the disk.

1.1 Project Overview

The project includes basic features of the file system like

1. Create file with specific permissions (Readonly, Read/Write, Execute)
2. Delete file
3. Write file
 - a. Append
 - b. Overwrite
4. Read file
5. Change File permission
6. Rename
7. Seeking free block in constant time

We will be devoting 20 MB of memory for implementing all these features. We will be using queue, treemap and priority queue to enhance the efficiency of all the features. We used java as the programming language to implement this file system.

1.2 Project Purpose

The main purpose of doing this project is to understand the implementation details of the file system and try to enhance the disk performance by using some algorithm for free system management. During the project, we will learn to implement basic file system calls like allocation of memory for file creation, opening file with different types of permissions (like read only, read write, read write and execute) and closing the file, deletion of file. Apart from this it will help to develop an insight regarding disk performance and disk space management.

2. Project Description

Generally, Operating systems have block size of 1-4KB. But here as mentioned, we are creating application using only 20 MB, So We will be dividing 20 MB memory in 163840 blocks, each of 128 bytes.

As we increase the size of the block, more and more fragmentation occurs. As we decrease the size of block the size of the File Allocation table will increase, which will cause wastage of memory without storing any useful information. It will also increase the time for seeking the free block, time for finding a particular file for read, write and rename. So, Small blocks are bad for performance but good for disk space utilization and larger blocks are bad for disk utilization but good for performance. So the size of 128 bytes per block is chosen to balance both the memory efficiency and time efficiency.

In order to store file location and access it, system will maintain two tables.

1. File Allocation Table
2. File Detail Table

2.1 File Detail Table

This table will store all the attributes of file. All the attributes are given prespecified space in File detail Table. Each attribute and its required memory are listed below.

Attribute No	Attribute Name	Size (Bytes)
1	File Name	10
2	Date and time of creation	12
3	Date and time of last modification	12
4	File size	3
5	File permissions	1
6	Block No from which file starts	3
7	End of the file in last block	1

Here because of memory limitation we are allowing only 13250 files entries in this table. And each file entry will consume 42 bytes of memory. So total memory consumed by this File Detail Table is $13250 \times 42 = 556500$ bytes. which is nearly 544 KB.

2.2 File Allocation Table

Any file of size more than 128 bytes will consume more than one memory block. To avoid external fragmentation we are not restricting memory allocation to continuous memory blocks. To overcome this limitation we are maintaining FAT, which will store the address of the next block of the particular file.

Each entry in FAT will consume 3 bytes of memory. For 20 MB space we made $(20 \times 1024 \times 1024 / 128) = 163840$ blocks of size 128 bytes. So the total size of FAT will be $163840 \times 3 = 491520$ bytes which is around 480 KB ($491520 / 1024 = 480$).

2.3 Features and its commands

1. Create file

- Command: creatf <filename>

This command will create the file with given filename. It will also ask for the permissions for the files like (Readonly, Read/Write, Execute). It will not create file if the any file with the same filename is existing.

2. Delete file

- Command: delf <filename>

This command will delete the file 'filename' if it is existing.

3. Read file

- Command: readf <filename>

This command will read the file 'filename'.

4. Write file

- Command: writf append <filename>

This command will append all the text to the existing file.

- Command: writf overwrite <filename>

This command will overwrite the text in existing file.

In both the command whenever user want to finish writing it need to write ****exit**** in new line which shows end of the file.

5. Change file permission

- Command: cpf <filename>

This command will ask the user to change file permissions.

6. Rename file

- Command: renamef <oldfilename> <newfilename>

This command will change the name of the file 'oldfilename' to 'newfilename'.

7. Listing all the files

- Command: list

This command will list down filenames of all the existing files in our virtual filesystem.

8. Listing details of all files

- Command: list l

This command will list down all the details of existing files in our virtual filesystem. It will give date and time of creation, date and time of last modification, file size, file permissions, and filename for all the files.

9. Details of a particular file

- Command: list <filename>

This command will give all the details of the file 'filename'. Like date and time of creation, date and time of last modification, file size, file permissions, and filename.

3. Optimization

Here we have focused on optimization of two features.

1. Seek time for free block

To keep track of all the free blocks we are maintaining a simple queue. Which will give a free block for allocation in constant time or in $O(1)$. Because FAT does not restricting the allocation of block to only continuous blocks, system does not need to traverse the whole memory for seeking the free block.

2. Reducing memory fragmentation

If a file need 10 blocks of memory then it will be accommodated even if there is no 10 continuous blocks available. Thus it will completely cancel the external fragmentation. To reduce the internal fragmentation, we have reduced the block size significantly to 128 bytes.

4. Future Scope

For now, this file system is implemented in one directory system. But it can be converted into multiple directory system by nominal modification. We can still reduce the internal fragmentation by some free space management system.

5. References

- 1.Modern Operating System by Andrew S. Tanenbaum
- 2.<http://pages.cs.wisc.edu/~remzi/OSTEP/>