CS342 OPERATING SYSTEMS

PROJECT #4

FAT32 File System Image Modifier

EXPERIMENTS REPORT

14.05.2024

Yasemin Akın Section 002 22101782

Göktuğ Serdar Yıldırım Section 002 22103111

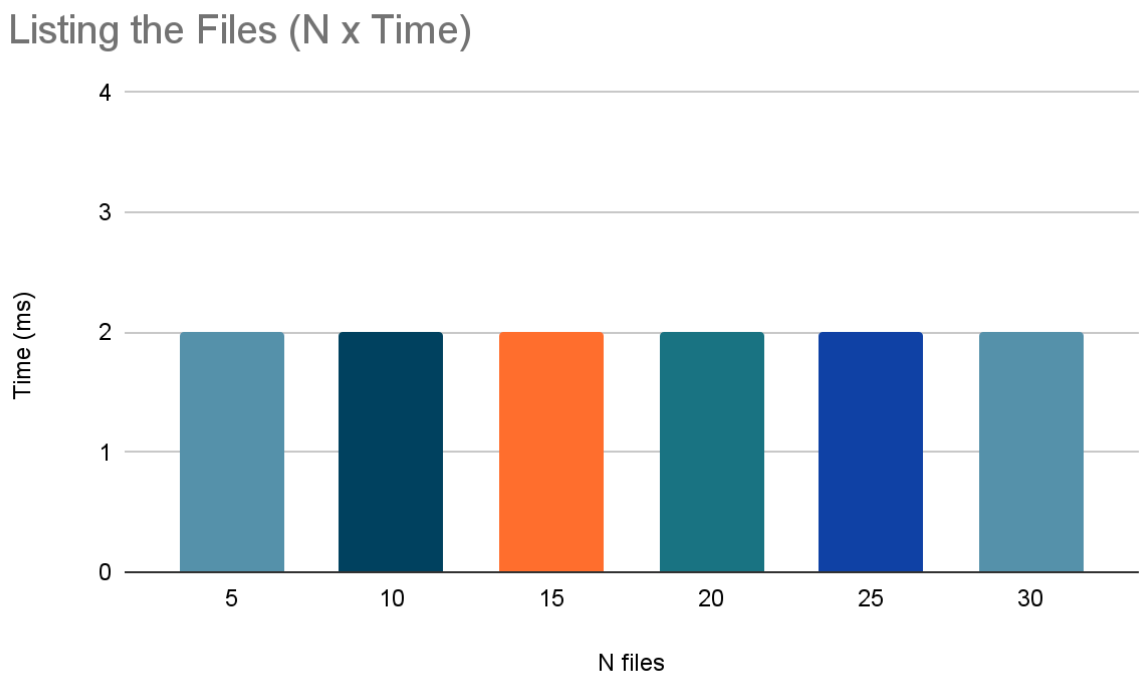
Introduction

The project FAT32 File System Image Modifier covers methods for comprehending and working with the FAT32 file system. The primary objective of the project is to create the software “fatmod”, which can read and alter data from a FAT32 disk image that is saved as a standard Linux file. The various basic activities of a file will be covered in this like file listing, file reading and writing, and managing directories and file information. Experience with file systems, raw disk access, and other FAT32 file system quirks will be gained from this project. Furthermore, we had an opportunity to work with the disk image at the byte level, which helps us understand how a file system organizes and preserves data.

Several experiments were conducted to determine the time required for file system activities, including listing files, creating and writing files, reading files, and deleting files, in order to assess the effectiveness of the FAT32 File System Image Modifier. In each trial, the base number of files that this function represented was changed. For the purpose of examining how the program scales with the incrementation of data, each experiment was conducted, meaning that the number of files varied between 5 and 30 (due to the constraint of the project which is maximum 31 files). This kind of experiment is helpful in determining the system's operational efficiency and potential areas for optimization. Among these, one should describe how different file operations behave in relation to the overall performance of the FAT32 file system, especially when the number of data increases.

Listing the All Files Exist in Disk

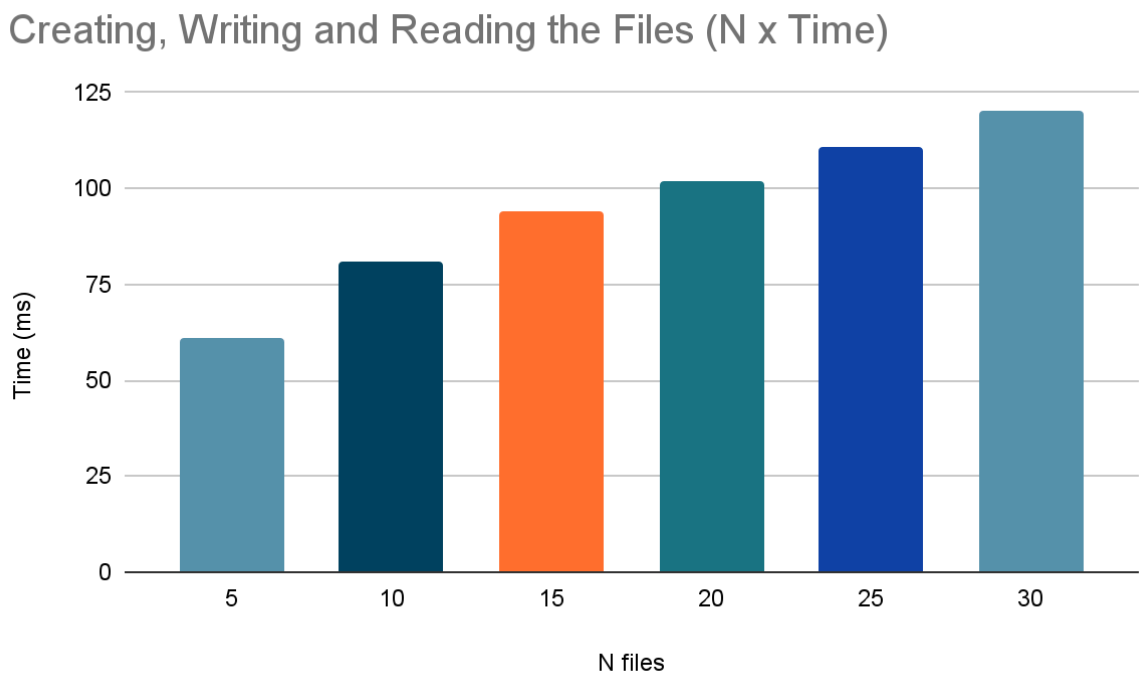
File Count	Time (ms)
5	2
10	2
15	2
20	2
25	2
30	2



The experiment focuses on the time spent on file listing in relation to the various file counts which range from five to thirty in the FAT32 filesystem image. Listing the files within the FAT32 filesystem images demonstrates that the time stays consistent in this example, it is 2 milliseconds and that this procedure takes that amount of time. In other words, the listing process in the FAT32 file system is streamlined, has little overhead, and works effectively for small to medium file counts. The performance consistency reveals that the overhead associated with listing files in the FAT32 file system is, at most, flat, insignificant, and regardless of the total number of files in the scope as indicated. The experiment demonstrates that the file listing's operation performs steadily even with small data sets and ensures dependable and quick performance. This could be because the FAT32 File System Image Modifier's file listing function is implemented well and the file system is easily constructed.

Creating, Writing and Reading the All Files Exist in Disk

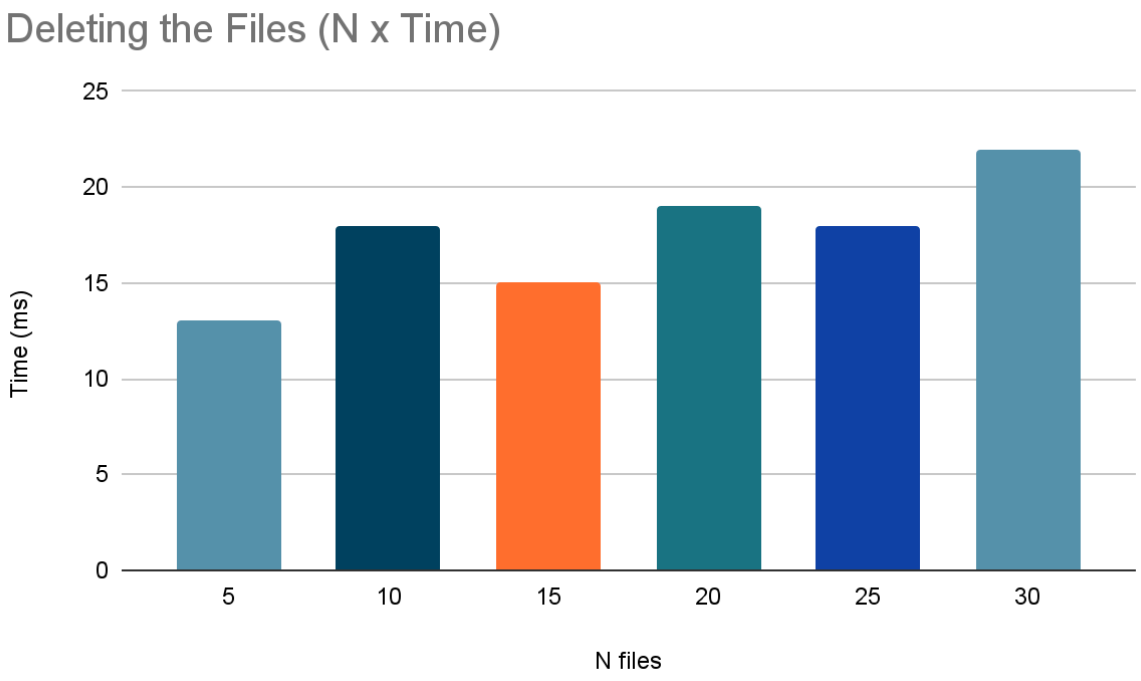
File Count	Time (ms)
5	61
10	81
15	94
20	102
25	111
30	120



This experiment displays all of the file create, write, and read outcomes. It is evident that as the number of files created increases from five to thirty: It took 61 milliseconds for 5 files, 81 milliseconds for 10 files, 94 milliseconds for 15 files, 102 milliseconds for 20 files, 111 milliseconds for 25 files, and 120 milliseconds for 30 files. The fact that the time grows very gradually was expected because, in general, the more files there are, the more time-consuming the processes become. The increase in the number of files can be used to explain this impact by adding overhead to the management of data blocks and file metadata. Despite the fact that these additions are relatively minor, they highlight the overall impact of handling the system's growing file count. Its complexity increases linearly with the number of files.

Deleting the All Files Exist in Disk

File Count	Time (ms)
5	13
10	18
15	15
20	19
25	18
30	22



The FAT32 experiment in deleting files has fluctuations in the time trend when there are file counts ranging from 5 to 30. At 10 files, deleting 5 files takes 13 milliseconds; at 15, it drops to 15 milliseconds. The duration climbs to 19 milliseconds for files 20 through 25, then drops to 18 and rises to 22 milliseconds for files 30 through 40. This does seem to point toward a small variation in file deletion timings that normally rises when more files are deleted. The FAT32 file system's internal procedures, which update the deallocated clusters and file allocation table, may be the cause of the modest variability. The system is not experiencing any overhead at extremely low file counts, but as the file count rises, the overhead becomes noticeable, as seen by an increase in deletion time with the increase in file count. Overall, the FAT32 file system modification performs flawlessly when it comes to deletions, maintaining a pretty steady performance level even when the number of files increases.

Conclusion

Regarding the file system's performance and scalability, the FAT32 File System Image Modifier experiments were quite on point. Nice observations are gained from the experiments on the FAT32 File System Image Modifier's performance in various activities. The file listing function seems to work rather well as a first experiment; the time is maintained at 2 milliseconds for all file counts, ranging from 5 to 30. This claims that the files are listed quickly and adaptably to the volume of small to medium sized files. When the number of files went from five to thirty, the time climbed monotonically. Due to the additional complexity and overhead associated with managing more files, the duration increased from 61 milliseconds for 5 files to 120 milliseconds for 30 files. The last experiment, which involved timing the deletion of files, revealed a time that increased gradually but generally: for five files, it took 13 milliseconds, while for thirty files, it took 22 milliseconds. According to this, even if the deletion process is effective for lower file counts, the overhead becomes apparent as file counts rise. According to all of these tests, the FAT32 File System Image Modifier works well for datasets that are lower in size, but when more files are added, the overhead increases. By knowing this, file system operations can be improved to maintain performance as data volumes increase, helping to better grasp the advantages and disadvantages of the FAT32 file system.