International Journal of Internet, Broadcasting and Communication Vol.16 No.4 402-408 (2024)
http://dx.doi.org/10.7236/IJIBC.2024.16.4.402

IJIBC 24-4-46

Design of a System to Initialize All Blocks within a Specific Folder

Seung Ju Jang[†]

[*†]*Professor, Department of Computer Engineering, Dong-Eui University, Busan, 47340 Korea*
sjjang@deu.ac.kr

Abstract

Initializing blocks of a file is a frequently used function on computers. One of the simplest ways to initialize a file block is to open the file and initialize all data. This allows you to completely erase the data the file previously contained. In this paper, we design a system that initializes file blocks within a specific folder. When you specify a folder you want to initialize, the files in the folder are found, the file data is read, and the read data is initialized and saved in the file. In this way, all file blocks in a specific folder are initialized. The experiment was performed on the function proposed in this paper. As a result of the experiment, it was confirmed that initialization of file blocks in a specific folder worked normally.

Keywords: *specific folder, block format, file initialization, security*

1. Introduction

The file system is a core component that stores and manages data in a computer. The file system stores files by dividing them into small units called blocks, and these blocks are the basic storage units of the file system. When a file is deleted or modified, the blocks of the file are initialized and the previous data is completely erased. However, for security reasons, there are times when the data of a specific file must be completely deleted or initialized. For example, if the blocks of the file are not completely initialized after deleting a file containing important personal information or confidential documents, there is a risk of data leakage.

To solve this problem, an effective method to initialize the blocks of a specific file is needed. A classic method for this is to delete the file and then perform an overwrite operation. However, this method makes all blocks unreliable and may remain in a recoverable state. Therefore, a safe and efficient file initialization method is needed.

In this paper, we propose a function to initialize the blocks of a specific file. The existing file deletion block structure is as shown in Figure 1 below.

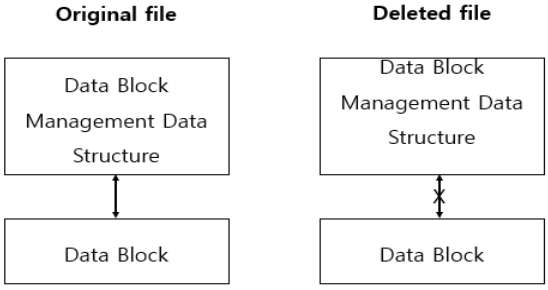

Figure 1. Existing deleted file block structure

Manuscript Received: October. 24, 2024 / Revised: October. 29, 2024 / Accepted: November. 4, 2024
Corresponding Author: sjjang@deu.ac.kr
Tel: +82-51-890-1710, Fax: +82-51-890-2629
Professor, Dept. of Computer Engineering, Dongeui University, Korea

As shown in Fig. 1, it generally shows the data block structure of the original file. In general, the data block of the original file consists of a data structure that manages the data block and a data block where the actual data exists. When a user deletes such a file, the operating system often does not delete the data block. In other words, as shown in the right figure of Fig. 1, When a user requests to delete a file, as shown in the right picture of Fig1, in many cases only the data structure that manages the data blocks contained in the file is deleted. That is, the data blocks that the file contains remain unchanged.

This existing file deletion method can cause various problems. Since the data block remains unchanged, it is possible to recover the data block if it is to be recovered. In most cases, the user expects the data block of the file to be deleted irreversibly and completely, but this does not happen.

In order to improve the problems of this existing file deletion method, this paper proposes a new file deletion method. The safety and efficiency of the proposed algorithm were evaluated and experiments were conducted by applying it in practice. This paper proposes a secure file initialization method to strengthen file security and personal information protection functions for files. The composition of this paper is explained in the following order: Chapter 2 describes related research, Chapter 3 describes system design, Chapter 4 describes experiments, and Chapter 5 describes conclusions.

2. Related research

File security and privacy are important issues in modern information technology. Even if a file is deleted from a computer system, the data of the file may not actually be removed and may still remain on the disk. This may lead to security issues such as data leakage. To solve these issues, active research is being conducted on how to initialize blocks held by specific files.

Recent studies have proposed various approaches and technologies for file initialization methods. These studies perform file initialization considering both safety and efficiency. One study proposed a method to identify the clusters that the file actually used after deleting a file and initialize the clusters [1, 2, 3]. This method guarantees complete deletion of data by directly initializing the clusters that the file used.

Another study proposed a method to randomly fill the blocks of a file using a specific algorithm and initialize them repeatedly when deleting a file [4, 5, 6]. This method makes it difficult to recover previous data even if the file is deleted. By repeating the process of filling the blocks randomly, it is difficult to regenerate previous data. In addition, some studies have proposed a method to completely overwrite data by writing random patterns on the disk when deleting a file. This method not only completely initializes the blocks used by the file to prevent data recovery, but also makes it difficult to identify or recover previous data by overwriting data using random patterns. These studies have presented new perspectives and approaches to file initialization methods to improve data security and privacy [4, 5, 6].

Existing file system initialization methods use various methods and techniques to erase and initialize file data. The commonly used initialization methods are as follows: Block data overwriting: The most basic file initialization method, which deletes a file and then overwrites it with random data or patterns. This method initializes all blocks of the file and erases previous data, but some blocks may not be completely initialized because the file is overwritten block by block [7, 8, 9]. Formatting: This method initializes the file system by formatting the disk. This method completely initializes the file system and erases all data, but since it initializes the entire disk, other data in the system may also be erased. Using a format tool (Tool-based): This method securely deletes files using a professional file deletion tool or disk utility. These tools use advanced algorithms to securely initialize all blocks of a file and make them unrecoverable [10, 11]. Encryption before Disposal: This method encrypts files before deleting them to keep them safe or delete them. This method reduces the

risk of data leakage because the files remain in encrypted form even after they are deleted. Physical Destruction: This method completely erases data by destroying the physical disk where the files are stored. This method is one of the most secure methods because it destroys data using a physical device.

3. System Design

The existing file deletion mechanism deletes a specified file from the file system. The file deletion mechanism mainly consists of the following processes: File metadata modification: The file system modifies the metadata of the file to delete the file. In this process, information such as the file name, path, size, and modification date are deleted. In addition, metadata for the block used by the file is also updated. Block release: The file system releases the block used by the file. In this process, the block is marked or initialized so that it can be used again. However, the data previously used may not be completely deleted and may remain in the block.

In order to design a method to initialize the block of a file, several factors must be considered. These factors should be designed considering the safety, efficiency, and applicability of the initialization method. In addition, a specific goal for the data block initialization method must be set. In this paper, this goal is to completely erase the previous data of the file and initialize it safely. In addition, the initialization process must be efficient and utilize system resources effectively.

In order to achieve this design goal, it is necessary to examine the problems of the existing block initialization method. The existing block initialization method generally initializes the entire file system. This method performs initialization for the entire block of the file system. In general, existing systems perform formatting on a file system basis. This system is not effective when only a specific file is irreversibly deleted.

This feature aims to completely wipe the blocks used by a file when it is deleted, making previous data unrecoverable. The function of irreversibly initializing blocks that a specific file has plays an important role in safely deleting data in a file system and protecting sensitive information from being leaked. This function aims to completely initialize blocks used by a file when the file is deleted, making it impossible to recover previous data.

For this purpose, the following functions are required: Block identification: First, the block used by a specific file must be identified. The file system can identify the block by checking the metadata of the block used by the file. Block initialization algorithm: The initialization algorithm is used to safely initialize blocks used by a specific file. This algorithm completely erases previous data and makes it impossible to recover by randomly overwriting the block or filling it with a random pattern. Guaranteeing irreversibility: It is important to check whether the data stored in the block has been completely deleted or changed to an irreversible state through the initialization process. This ensures irreversible initialization of the block. Security: The block initialization function should also be reliable in terms of security. The initialized block should be safely protected, and the block initialization process itself should not be vulnerable to security threats such as hacking or data leakage.

A block initialization function that satisfies these requirements can play an important role in strengthening data security and privacy protection in the file system. Therefore, various aspects such as safety, efficiency, and security should be considered in the design and implementation of this function. The operation process of the function to format a file block in a specific folder proposed in this paper is as shown in Fig. 2.

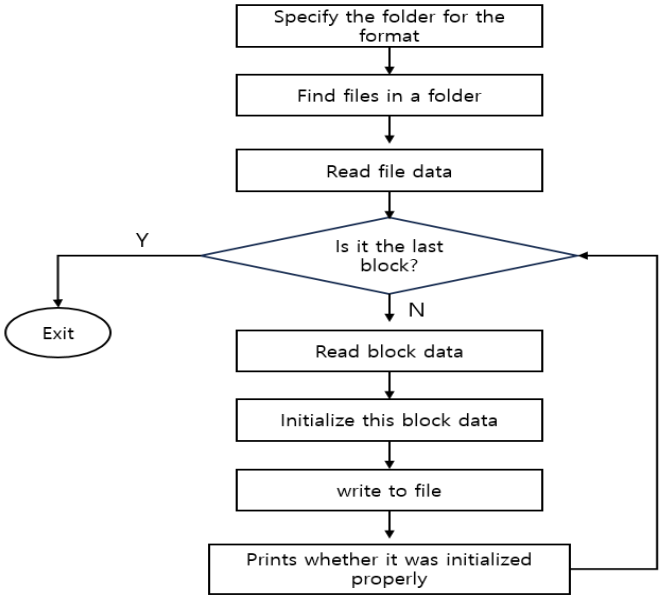

Figure 2. Process of initializing a file block in a specific folder

In order to format a file block in a specific folder, you must specify the folder you want to format. Once the folder you want to format is specified, the files in the folder are searched for. For the found files, the data (blocks) in the file are read into the main memory space. If it is not the last block in the file, the block is read into the main memory space and this block is initialized. The initialized block data is saved back to a file. A task is performed to check whether the initialization was done properly for the initialized block. This process is repeated until there are no more blocks to read in the file. When there are no more blocks to read in the file, the program execution is terminated.

4. Experiment

An environment was built to test whether the function of formatting a file block in a specific folder proposed in this paper works properly. The experimental environment used the Windows 11 operating system and the CodeBlock compiler. A program was written to designate a specific folder using the CodeBlock compiler and initialize the files in the designated folder. The screen for testing the function of designating the blocks of files in a specific folder using the written program is as shown in Fig. 3 below. Fig. 3 is the screen for designating a specific folder to be formatted.

Figure 3. Screen for designating a specific folder

Fig. 3 is the screen for designating a folder to be formatted (initialized). In this paper, the "C:\Temp\ 6666666" folder was designated. Fig. 3 shows the files in the "C:\Temp\6666666" folder.

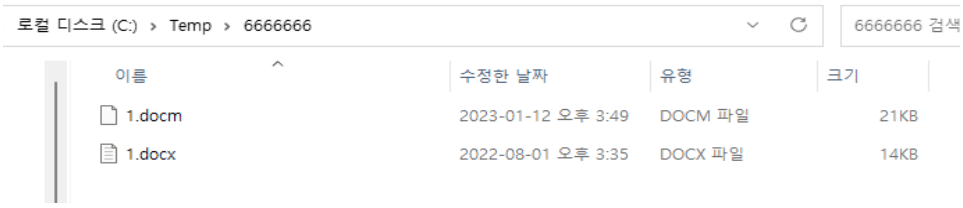

Figure 4. Files in a specific folder

There are two files, “1.docm” and “1.docx”, in the “C:\Temp\6666666” folder. The following Fig. 4 is a screen that checks whether the data block is formatted normally after reading the file in a specific folder and formatting (initializing) it.

Figure 5. Initializing a file in a specific folder

Fig. 5 shows the process of finding and formatting a file in the “C:\Temp\6666666” folder. Fig. 5 shows the process of formatting (initializing) the “1.docm” file in the “C:\Temp\6666666” folder. It shows that all the block data in the “1.docm” file are formatted to 0.

The following Fig. 6 shows the process of formatting (initializing) the “1.docx” file in the “C:\Temp\6666666” folder.

Figure 6. Initializing a file in a specific folder

Fig. 6 shows the process of formatting (initializing) the “1.docx” file in the “C:\Temp\6666666” folder. It shows that all the block data in the “1.docx” file are formatted to 0. After going through this process, the following Fig. 6 shows the folder appearance after formatting the “1.docm” and “1.docx” files in the “C:\Temp\6666666” folder.

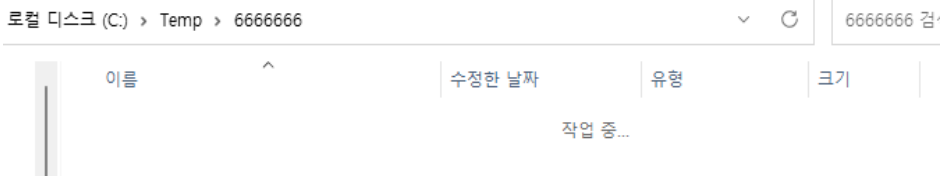

Figure 7. End of initialization of file blocks in a specific folder

Fig. 7 shows the folder appearance after formatting the "1.docm" and "1.docx" files in the "C:\Temp\6666666" folder. After formatting the "C:\Temp\6666666" folder, you can see that all the files in the folder have disappeared normally. This means that all the blocks in this folder have been initialized, and you can see that the files have disappeared.

In this paper, we conducted an experiment on the format (initialization) process for block data contained in files within a specific folder. As a result of the experiment, we were able to confirm that the initialization was performed normally for the blocks in the files in the specific folder.

5. Conclusion

In this paper, we designed a function to format (initialize) data blocks of files in a specific folder in the Windows operating system. This function first specifies a specific folder. When a folder is specified, information about files in the folder is read. For each file, data is read into the main memory in block units and then initialized to 0. The data initialized to 0 is overwritten with the block data of the original file. This process is performed until there are no more file blocks.

After performing block initialization on files in a specified folder, we can confirm that the blocks were initialized normally through an experiment to confirm that they were initialized normally. We conducted an experiment to confirm whether the block initialization function for files in a specified folder in the Windows operating system proposed in this paper operates normally. As a result of the experiment, we were able to confirm that the files in the specified block were initialized (formatted) to 0 normally.

References

[1] S. W. Lee, Se-Jin OH, "Index management technique using Small block in storage device based on NAND flash memory," Journal of The Korea Society of Computer and Information, Vol. 25 No. 10, pp. 1-14, October 2020. DOI:https://doi.org/10.9708/jksci.2020.25.10.001.

[2] Dmytro Ostrovka, and Vasyl Teslyuk, "The Analysis of File Format Conversion Tools for Storing 3D Objects for the iOS Platform", Lviv Polytechnic National University, 12 Bandera street, Lviv, Ukraine, 79013. 2020.

[3] Kazuki Onishi; Jaehoon yu; Masanori Hashimoto, "Memory Efficient Training using Lookup-Table-based Quantization for Neural Network", 2020 2nd IEEE International Co, 31 August 2020 - 02 September 2020, IEEE Xplore: 23 April 2020. DOI: https://doi.org/10.1109/AICAS48895.2020.9073989.

[4] Christian Hummert, Dirk Pawlaszczyk, Mobile Forensics – The file format handbooks, 2022, DOI:https://doi.org/10.1007/978-3-030-98467-0.

[5] Michal Kvet, "Database Block Management using Master Index", 2022 32nd Conference of Open Innovations Association (FRUCT), 09-11 November 2022, DOI: https://doi.org/10.23919/FRUCT56874.2022.9953806.

[6] Jui-Nan Yen; Yao-Ching Hsieh; Cheng-Yu Chen; Tseng-Yi Chen; Chia-Lin Yang; Hsiang-Yun Cheng; Yixin Luo, "Efficient Bad Block Management with Cluster Similarity", 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), 02-06 April 2022, DOI: https://doi.org/10.1109/HPCA53966.2022.00044.

[7] G. Yadgar, M. Gabel, S. Jaffer and B. Schroeder, "SSD-based workload characteristics and their performance implications", ACM Transactions on Storage (TOS), vol. 17, no. 1, pp. 1-26, 2021, DOI: https://doi.org/10.1145/3423137.

[8] Fakhitah Ridzuan, Wan Mohd Nazmee Wan Zainon, "A Review on Data Cleansing Methods for Big

Data", DOI: https://doi.org/10.1016/j.procs.2019.11.177.

[9] Yingzhe He, Guozhu Meng, Kai Chen, Jinwen He, Xingbo Hu, "DeepObliviate: A Powerful Charm for Erasing Data Residual Memory in Deep Neural Networks", arXiv:2105.06209 [cs.LG] (or arXiv:2105.06209v1 [cs.LG] for this version), DOI: https://doi.org/10.48550/arXiv.2105.06209.

[10] Harish Dattatraya Dixit, Sneha Pendharkar, Matt Beadon, Chris Mason, Tejasvi Chakravarthy, Bharath Muthiah, Sriram Sankar, "Silent Data Corruptions at Scale", arXiv:2102.11245 [cs.AR] (or ar Xiv:2102.11245v1 [cs.AR] for this version), DOI: https://doi.org/10.48550/arXiv.2102.11245.

[11] Dong-Jin Shin and Jeong-Joon Kim, "Cache-Based Matrix Technology for Efficient Write and Recovery in Erasure Coding Distributed File Systems", 2023, 15(4), 872; DOI: https://doi.org/10.3390/sym15040872.