

# CORE:

## Custom OS-level Resource Emulator

Richmonde Paulo G. Gatchalian, Lauren Frances M. Salazar, Ralph Christian B. Tejada

Department of Electronics and Computer Engineering  
Gokongwei College of Engineering  
De La Salle University Manila

2401 Taft Avenue, Malate, Manila, Philippines 1004

[richmonde\\_gatchalian@dlsu.edu.ph](mailto:richmonde_gatchalian@dlsu.edu.ph), [lauren\\_salazar@dlsu.edu.ph](mailto:lauren_salazar@dlsu.edu.ph), [ralph\\_tejada@dlsu.edu.ph](mailto:ralph_tejada@dlsu.edu.ph)

**Abstract**—File systems are fundamental to modern computing, governing how data is stored, retrieved, and managed. This paper presents the design and implementation of a custom file system built using C and shell scripting, operating on a virtual disk. The system simulates core file operations such as creation, deletion, viewing, editing, and listing of files while using contiguous allocation for storage management. Implemented with a block-based virtual disk image, the system demonstrates how real-world file systems like FAT and ext4 handle file allocation and space utilization. It features a command-line interface and a basic file entry structure, offering a simple yet functional environment for understanding underlying OS concepts. By working directly with binary data and simulating block-level access, this project offers valuable insights into virtual file storage and allocation mechanisms. The simplicity of the design makes it an educational tool for students studying operating systems or systems programming.

**Keywords** — File System, Virtual Disk, File Allocation, Contiguous Allocation, C Programming, Operating Systems

### I. INTRODUCTION

File systems are fundamental components of modern operating systems, enabling structured storage, efficient access, and reliable management of data on both physical and virtual devices. As digital systems evolve, the need for lightweight, secure, and transparent file storage frameworks becomes more apparent, particularly in environments where abstraction and scalability often obscure the core mechanisms of data handling. This paper presents CORE: Custom OS-level Resource Emulator, a simplified file system developed in C and shell scripting that emulates key OS-level storage behaviors on a 1MB virtual disk image. CORE implements basic file operations such as creation, deletion, listing, and editing, using a contiguous allocation scheme to simulate low-level storage interactions. Contemporary work in embedded and cloud systems reflects similar goals. MIFS, a flash file system for constrained devices, introduces B+ tree indexing over continuous memory to improve performance and reduce memory use [1]. BlockPres applies blockchain-integrated IPFS storage for medical data, demonstrating secure, decentralized file control [2]. In large-scale systems, Chevron’s data platform shows the importance of scalable and structured data delivery for decision-making across industrial infrastructures [3]. Research on vPIM explores performance optimization in virtualized memory environments, offering insights into low-overhead system design [4]. Meanwhile, SigmaOS unifies microservice and serverless workloads through a minimal, efficient OS structure, emphasizing the growing demand for flexible, OS-level abstractions [5]. CORE aims to bridge these ideas with an educational and modular design that exposes the inner workings of file allocation, metadata tracking, and block-based storage through direct interaction, offering a transparent learning tool for operating system fundamentals.

### II. BACKGROUND OF THE STUDY

File systems are essential components of operating systems, responsible for managing how data is stored, accessed, and organized within storage media. They provide the structure for handling both file content and metadata, ensuring that users and applications can interact with stored information reliably. Widely used systems such as FAT, NTFS, ext4, and HFS each follow different design philosophies, particularly in how they allocate space and track file states. For example, FAT uses simple linked allocation, while NTFS incorporates a Master File Table (MFT) and journaling to enhance recoverability and metadata tracking [6]. Understanding these internal mechanisms is critical not only for systems programming but also for fields such as digital forensics, where the ability to analyze deleted, modified, or fragmented files depends on knowledge of file system architecture [7]

Recent research has demonstrated how forensic analysis tools reconstruct event timelines by parsing NTFS metadata. These tools extract values such as event identifiers and user session information to establish behavioral traces in a system.

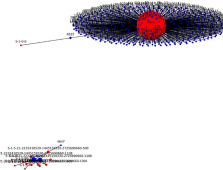


Fig. 1. Graph showing relationships between NTFS event\_id and user\_sid attributes, illustrating how metadata links user actions to system events [8]

Additionally, timestamp metadata, such as Modified, Accessed, Changed, and Birth (MACB) values are crucial for identifying patterns in file activity and anomaly detection.

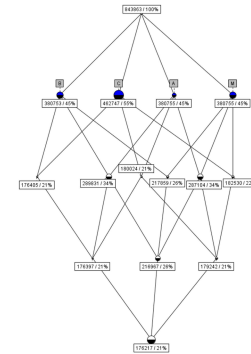


Fig. 1. Concept lattice for MACB timestamps in NTFS data, illustrating frequency and combinations of file system metadata events [8].

Complementary studies also show the value of simplified and configurable operating system environments for education and experimentation. A user-friendly kernel that includes essential features such as memory handling, syscalls, and file systems can help students better understand the role and behavior of core OS components [9]. CORE: Custom OS-level Resource Emulator builds on these ideas, offering a hands-on platform that simulates file allocation, metadata management, and virtual storage using a block-based approach to help users grasp the core principles of file system design.

### III. STATEMENT OF THE PROBLEM

Modern advancements in storage hardware have significantly outpaced the evolution of traditional software stacks, creating inefficiencies that limit system performance and transparency. Although solid-state drives (SSDs) and protocols like NVMe and SAS offer high-speed data access, legacy storage architectures continue to introduce delays and obscure file system behavior [10]. This disconnect makes it difficult for learners and developers to fully understand how storage systems function at a low level.

At the same time, growing concerns over data reliability and loss, particularly in distributed and cloud environments, highlight the risks of relying on abstracted storage layers. Systems that implement encryption and multi-file redundancy demonstrate the importance of having visibility into file allocation and storage processes [11]. CORE: Custom OS-level Resource Emulator was developed in response to these challenges. It provides a simplified, transparent platform that allows users to interact directly with file operations and block-level allocation through a virtual disk.

### IV. SIGNIFICANCE OF THE STUDY

Understanding file systems at the block level is a foundational skill in systems programming, cybersecurity, and computer engineering. Despite their importance, modern operating systems often obscure the inner workings of file allocation, metadata tracking, and low-level storage interaction. As a result, students and entry-level developers are frequently introduced to file systems only through high-level APIs, with limited exposure to how data is actually structured and managed. CORE: Custom OS-level Resource Emulator bridges this gap by offering a lightweight, emulator-based environment where users can directly interact with virtual storage, simulating key file operations such as creation, deletion, and content editing using contiguous block allocation.

Projects like forkSim, which visualizes process-level concurrency in operating systems education, illustrate the growing value of simulation-based tools for teaching technical concepts [12]. Similarly, CORE provides a structured, interactive platform for learning file system fundamentals. Its role is comparable to Deimos, an educational operating system developed for robotics platforms, which demonstrates the importance of making OS-level tools accessible for students and researchers [13]. In parallel, the rising interest in decentralized file storage architectures such as IPFS and Web3 underscores the continued need for foundational knowledge in traditional storage systems [14]. CORE supports this learning by simulating the behaviors and constraints that define real-world file systems, encouraging a deeper understanding of how operating systems manage data behind the scenes.

### V. OBJECTIVES

To develop an intuitive and interactive emulator, CORE, which simulates a basic operating system's file allocation process, providing an educational tool for students to visualize and understand file system behavior and resource management. The following are the key objectives:

1. Design and implement a virtual disk image within the CORE emulator to simulate file storage and allocation using contiguous block allocation.
2. Develop user-friendly interfaces that allow users to interact with the emulator by creating, viewing, deleting, and editing files, offering real-time feedback on file system operations.
3. Simulate core file system behaviors, such as allocation, deletion, and reallocation, to help students visualize the internal workings of file systems and resource management.
4. Provide clear documentation and guides within the emulator to assist students in understanding the impact of different file operations on storage, file fragmentation, and space utilization.
5. Test the emulator with students and assess its effectiveness in improving their comprehension of OS-level storage management, resource allocation, and file system structures.

### VI. DESCRIPTION OF THE SYSTEM

CORE: Custom OS-level Resource Emulator is a simplified file system emulator designed to simulate how operating systems manage storage at the block level. Developed in C and supported by a shell script interface, CORE operates on a 1MB virtual disk (virtual\_disk.img) and allows users to perform basic file operations such as creation, deletion, viewing, editing, and listing. The system uses a FileEntry structure to manage file metadata and a block\_map to track allocation across 4096 fixed-size blocks. All interactions with the virtual disk are handled through standard C file I/O functions, providing a clear and accessible view into how real file systems perform low-level data manipulation.

CORE uses contiguous allocation, where each file is stored in sequential blocks. This method is intentionally chosen for its simplicity and ease of visualization, allowing learners to better grasp concepts like block tracking, fragmentation, and allocation strategy. The architecture includes a shell-based interface (file\_system.sh) for initializing the virtual disk and compiling the system, while the main logic resides in file\_system.c. By stripping down modern file system complexity into a modular, transparent emulator, CORE serves as a hands-on educational tool that bridges the gap between theoretical OS concepts and real-world file system behavior.

## VII. METHODOLOGY

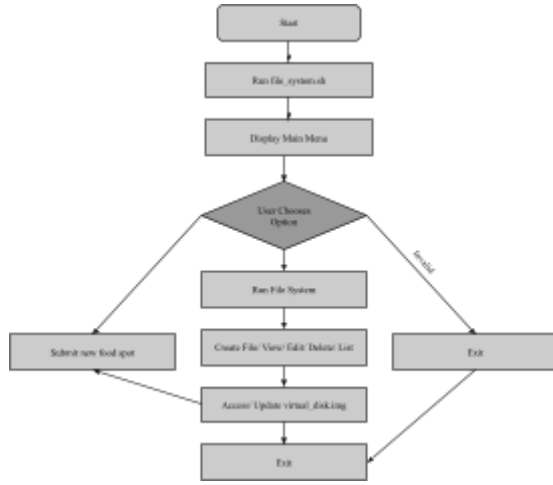


Fig. 3. System Flowchart

The development of CORE followed a structured, modular approach to emulate a working file system within a constrained virtual environment. The emulator was built using C programming for direct file manipulation and shell scripting to handle user interaction, compilation, and disk setup. A virtual disk image (virtual\_disk.img) was created to simulate a block-based storage system, with each block set to 256 bytes. The file system tracks allocation using a block map and manages metadata through a file entry structure stored in memory.

Users interact with the system through a simple terminal menu presented via a shell script. Upon selection, the appropriate C function is invoked to execute file operations such as creation, deletion, viewing, editing, or listing. For file creation, the system checks for duplicate filenames, calculates the required number of blocks, searches for free contiguous space, writes the content into the virtual disk, and updates the metadata. Similar structured logic applies for the other operations, with file data being accessed or modified directly through standard I/O functions like `fseek()` and `fwrite()`. All changes are immediately reflected in the virtual disk to simulate real-time persistence.

## VIII. REVIEW OF RELATED LITERATURE

The design and behavior of file systems have long been central topics in systems programming, yet their internal mechanisms often remain opaque to learners due to the abstraction provided by modern operating systems. This gap between theoretical understanding and practical application has driven researchers to develop solutions that improve transparency, performance, and accessibility. Several works have focused on enhancing storage performance through smarter allocation strategies and memory-aware designs. Yu et al. introduced VMAlloc, a wear-leveling-aware multi-grained allocator for persistent memory file systems. Their system significantly reduced uneven write distribution, which is a major cause of early memory degradation in PM systems [15]. In a similar direction, Liu and Wang proposed CFFS, a file system built around contiguous file allocation and fine-grained metadata to improve `mmap` performance and reduce fragmentation in high-speed memory environments [16]. These contributions highlight the value of optimizing memory allocation and reuse in block-based storage, a concept CORE seeks to simulate within an educational environment.

Beyond allocation techniques, researchers have explored fault-tolerant and power-safe systems for embedded and resource-constrained platforms. Munegowda et al. examined transaction-safe cluster allocation in the TexFAT system, designed for embedded Windows CE systems. Their study emphasizes the importance of file system behavior under restricted power conditions and limited resources [17]. Devulapalli et al. proposed integrating object-based storage devices into parallel file systems to improve efficiency and decentralize metadata management, which can reduce performance bottlenecks in distributed systems [18]. Peng et al. focused on addressing write amplification in SSDs by introducing PULSE, a log-structured enhancement for B-epsilon-tree storage models that reduces wear and improves throughput [19]. These studies collectively show that understanding allocation strategies and their real-world implications is crucial for system designers, and CORE can provide a foundation for such knowledge by simulating similar behaviors through basic contiguous allocation logic.

Recent advancements have also shown growing interest in decentralized storage systems. Blockchain-based frameworks, in particular, offer improved security, transparency, and control over data. Mishra et al. proposed a prototype for secure medical record management using blockchain and IPFS, providing a decentralized platform with smart contract support for access control and data integrity [20]. Khan et al. developed BlockPres, a system that uses IPFS to securely store patient prescriptions while ensuring access control for multiple healthcare providers [2]. Yang et al. contributed a ring signature scheme that allows for both anonymity and traceability, linking medical data across different providers while maintaining security and auditability [21]. Anand et al. combined blockchain and transfer learning to create TraVel, a framework for filtering and securing IoT data streams before storage on a private Ethereum network [3]. To improve scalability in such systems, Mythili and Gopalakrishnan designed an adaptive sharded blockchain protocol for healthcare data, significantly increasing throughput while maintaining low latency [22]. Although these systems are far more complex than CORE, they emphasize the long-term importance of understanding file and data storage structures, even at a fundamental level.

A few researchers have addressed the need for educational or simplified platforms. Poinot et al. presented a simulation framework for data management in distributed systems that uses a data-oriented model to optimize workflow and reduce redundant storage [Poinot]. On a more hybridized approach, Nguyen and Leis revisited the longstanding debate on storing files versus BLOBs in databases. They proposed a system that uses a lightweight FUSE layer to expose database-stored BLOBs as files, balancing transactional integrity with usability [23]. These systems, although aimed at enterprise or research environments, align with the goals of CORE by emphasizing visibility, direct access, and architectural clarity. CORE positions itself uniquely as a teaching tool by offering a simplified, hands-on experience with file creation, deletion, viewing, and metadata tracking. By making these processes transparent and accessible, the system provides learners with a direct understanding of how operating systems manage data at the most fundamental level.

## IX. THEORETICAL CONSIDERATIONS

The design of CORE draws from established theories in memory allocation, system optimization, and resource modeling. At its core, CORE implements a contiguous block allocation strategy, which reflects principles outlined by Duong and Hur in their study of translation lookaside buffer (TLB) prefetching. Their work demonstrated how exploiting memory contiguity at the block level improves I/O efficiency and reduces latency caused by page-table

walks, a concept that CORE simplifies to illustrate allocation behavior in a virtual environment [24]. In modeling resource interactions, CORE also aligns with the framework proposed by Samy et al., who used deep reinforcement learning to optimize task offloading and resource consumption in mobile edge computing. While CORE does not employ machine learning algorithms, it simulates static, deterministic resource allocation that mirrors foundational ideas in system-level decision-making under constrained conditions [25]. Additionally, the simulation's structure draws inspiration from the differential game-based optimization model discussed by Lv, where interactions between institutions and technologies were analyzed for optimal outcomes. CORE reflects a similar spirit by allowing users to interactively explore how changes in file size, placement, and deletion affect virtual disk space, allocation efficiency, and resource reuse [26]. These theoretical models collectively reinforce CORE's role as a minimalistic but pedagogically grounded emulator designed to demystify low-level file system operations through hands-on experimentation.

## X. DATA AND RESULTS

The group has developed the file system simulator in a virtual box using the Fedora Linux environment. The implementation of CORE successfully demonstrates key file system operations on a simplified virtual disk environment. The emulator's behavior aligns with expected outcomes based on the underlying principles of file system design using contiguous allocation.

```
liveuser@localhost-live:~$ chmod +x file_system.sh
liveuser@localhost-live:~$ ./file_system.sh
Custom File System
1. Initialize Virtual Disk
2. Run File System
3. Exit
Choose an option: 1
```

Fig. 4. Setting up and Shell Script Menu

First, `chmod +x` is used to make the shell script executable. The script is then run using `./`, which tells the system to execute the file from the current directory. Once executed, the script displays a menu that allows the user to either initialize a new virtual disk or launch the main file system. If a virtual disk already exists, the script will notify the user that it is ready for use. By default, the virtual disk is allocated 1 MB of storage for the emulator's file operations, though this size can be adjusted as needed.

```
Custom File System
1. Initialize Virtual Disk
2. Run File System
3. Exit
Choose an option: 1
10485760 records in
10485760 records out
10485760 bytes (1.0 MB, 1.0 MiB) copied, 0.51400 s, 161 MB/s
Virtual disk initialized: virtual_disk.img
liveuser@localhost-live:~$ ls
file_system  file_system.sh  pictures  tasks  tasks1  tasks2  tasks3  tasks4  tasks5  tasks6  tasks7  tasks8  tasks9  tasks10  tasks11  tasks12  tasks13  tasks14  tasks15  tasks16  tasks17  tasks18  tasks19  tasks20  tasks21  tasks22  tasks23  tasks24  tasks25  tasks26  tasks27  tasks28  tasks29  tasks30  tasks31  tasks32  tasks33  tasks34  tasks35  tasks36  tasks37  tasks38  tasks39  tasks40  tasks41  tasks42  tasks43  tasks44  tasks45  tasks46  tasks47  tasks48  tasks49  tasks50  tasks51  tasks52  tasks53  tasks54  tasks55  tasks56  tasks57  tasks58  tasks59  tasks60  tasks61  tasks62  tasks63  tasks64  tasks65  tasks66  tasks67  tasks68  tasks69  tasks70  tasks71  tasks72  tasks73  tasks74  tasks75  tasks76  tasks77  tasks78  tasks79  tasks80  tasks81  tasks82  tasks83  tasks84  tasks85  tasks86  tasks87  tasks88  tasks89  tasks90  tasks91  tasks92  tasks93  tasks94  tasks95  tasks96  tasks97  tasks98  tasks99  tasks100  tasks101  tasks102  tasks103  tasks104  tasks105  tasks106  tasks107  tasks108  tasks109  tasks110  tasks111  tasks112  tasks113  tasks114  tasks115  tasks116  tasks117  tasks118  tasks119  tasks120  tasks121  tasks122  tasks123  tasks124  tasks125  tasks126  tasks127  tasks128  tasks129  tasks130  tasks131  tasks132  tasks133  tasks134  tasks135  tasks136  tasks137  tasks138  tasks139  tasks140  tasks141  tasks142  tasks143  tasks144  tasks145  tasks146  tasks147  tasks148  tasks149  tasks150  tasks151  tasks152  tasks153  tasks154  tasks155  tasks156  tasks157  tasks158  tasks159  tasks160  tasks161  tasks162  tasks163  tasks164  tasks165  tasks166  tasks167  tasks168  tasks169  tasks170  tasks171  tasks172  tasks173  tasks174  tasks175  tasks176  tasks177  tasks178  tasks179  tasks180  tasks181  tasks182  tasks183  tasks184  tasks185  tasks186  tasks187  tasks188  tasks189  tasks190  tasks191  tasks192  tasks193  tasks194  tasks195  tasks196  tasks197  tasks198  tasks199  tasks200  tasks201  tasks202  tasks203  tasks204  tasks205  tasks206  tasks207  tasks208  tasks209  tasks210  tasks211  tasks212  tasks213  tasks214  tasks215  tasks216  tasks217  tasks218  tasks219  tasks220  tasks221  tasks222  tasks223  tasks224  tasks225  tasks226  tasks227  tasks228  tasks229  tasks230  tasks231  tasks232  tasks233  tasks234  tasks235  tasks236  tasks237  tasks238  tasks239  tasks240  tasks241  tasks242  tasks243  tasks244  tasks245  tasks246  tasks247  tasks248  tasks249  tasks250  tasks251  tasks252  tasks253  tasks254  tasks255  tasks256  tasks257  tasks258  tasks259  tasks260  tasks261  tasks262  tasks263  tasks264  tasks265  tasks266  tasks267  tasks268  tasks269  tasks270  tasks271  tasks272  tasks273  tasks274  tasks275  tasks276  tasks277  tasks278  tasks279  tasks280  tasks281  tasks282  tasks283  tasks284  tasks285  tasks286  tasks287  tasks288  tasks289  tasks290  tasks291  tasks292  tasks293  tasks294  tasks295  tasks296  tasks297  tasks298  tasks299  tasks300  tasks301  tasks302  tasks303  tasks304  tasks305  tasks306  tasks307  tasks308  tasks309  tasks310  tasks311  tasks312  tasks313  tasks314  tasks315  tasks316  tasks317  tasks318  tasks319  tasks320  tasks321  tasks322  tasks323  tasks324  tasks325  tasks326  tasks327  tasks328  tasks329  tasks330  tasks331  tasks332  tasks333  tasks334  tasks335  tasks336  tasks337  tasks338  tasks339  tasks340  tasks341  tasks342  tasks343  tasks344  tasks345  tasks346  tasks347  tasks348  tasks349  tasks350  tasks351  tasks352  tasks353  tasks354  tasks355  tasks356  tasks357  tasks358  tasks359  tasks360  tasks361  tasks362  tasks363  tasks364  tasks365  tasks366  tasks367  tasks368  tasks369  tasks370  tasks371  tasks372  tasks373  tasks374  tasks375  tasks376  tasks377  tasks378  tasks379  tasks380  tasks381  tasks382  tasks383  tasks384  tasks385  tasks386  tasks387  tasks388  tasks389  tasks390  tasks391  tasks392  tasks393  tasks394  tasks395  tasks396  tasks397  tasks398  tasks399  tasks400  tasks401  tasks402  tasks403  tasks404  tasks405  tasks406  tasks407  tasks408  tasks409  tasks410  tasks411  tasks412  tasks413  tasks414  tasks415  tasks416  tasks417  tasks418  tasks419  tasks420  tasks421  tasks422  tasks423  tasks424  tasks425  tasks426  tasks427  tasks428  tasks429  tasks430  tasks431  tasks432  tasks433  tasks434  tasks435  tasks436  tasks437  tasks438  tasks439  tasks440  tasks441  tasks442  tasks443  tasks444  tasks445  tasks446  tasks447  tasks448  tasks449  tasks450  tasks451  tasks452  tasks453  tasks454  tasks455  tasks456  tasks457  tasks458  tasks459  tasks460  tasks461  tasks462  tasks463  tasks464  tasks465  tasks466  tasks467  tasks468  tasks469  tasks470  tasks471  tasks472  tasks473  tasks474  tasks475  tasks476  tasks477  tasks478  tasks479  tasks480  tasks481  tasks482  tasks483  tasks484  tasks485  tasks486  tasks487  tasks488  tasks489  tasks490  tasks491  tasks492  tasks493  tasks494  tasks495  tasks496  tasks497  tasks498  tasks499  tasks500  tasks501  tasks502  tasks503  tasks504  tasks505  tasks506  tasks507  tasks508  tasks509  tasks510  tasks511  tasks512  tasks513  tasks514  tasks515  tasks516  tasks517  tasks518  tasks519  tasks520  tasks521  tasks522  tasks523  tasks524  tasks525  tasks526  tasks527  tasks528  tasks529  tasks530  tasks531  tasks532  tasks533  tasks534  tasks535  tasks536  tasks537  tasks538  tasks539  tasks540  tasks541  tasks542  tasks543  tasks544  tasks545  tasks546  tasks547  tasks548  tasks549  tasks550  tasks551  tasks552  tasks553  tasks554  tasks555  tasks556  tasks557  tasks558  tasks559  tasks560  tasks561  tasks562  tasks563  tasks564  tasks565  tasks566  tasks567  tasks568  tasks569  tasks570  tasks571  tasks572  tasks573  tasks574  tasks575  tasks576  tasks577  tasks578  tasks579  tasks580  tasks581  tasks582  tasks583  tasks584  tasks585  tasks586  tasks587  tasks588  tasks589  tasks590  tasks591  tasks592  tasks593  tasks594  tasks595  tasks596  tasks597  tasks598  tasks599  tasks600  tasks601  tasks602  tasks603  tasks604  tasks605  tasks606  tasks607  tasks608  tasks609  tasks610  tasks611  tasks612  tasks613  tasks614  tasks615  tasks616  tasks617  tasks618  tasks619  tasks620  tasks621  tasks622  tasks623  tasks624  tasks625  tasks626  tasks627  tasks628  tasks629  tasks630  tasks631  tasks632  tasks633  tasks634  tasks635  tasks636  tasks637  tasks638  tasks639  tasks640  tasks641  tasks642  tasks643  tasks644  tasks645  tasks646  tasks647  tasks648  tasks649  tasks650  tasks651  tasks652  tasks653  tasks654  tasks655  tasks656  tasks657  tasks658  tasks659  tasks660  tasks661  tasks662  tasks663  tasks664  tasks665  tasks666  tasks667  tasks668  tasks669  tasks670  tasks671  tasks672  tasks673  tasks674  tasks675  tasks676  tasks677  tasks678  tasks679  tasks680  tasks681  tasks682  tasks683  tasks684  tasks685  tasks686  tasks687  tasks688  tasks689  tasks690  tasks691  tasks692  tasks693  tasks694  tasks695  tasks696  tasks697  tasks698  tasks699  tasks700  tasks701  tasks702  tasks703  tasks704  tasks705  tasks706  tasks707  tasks708  tasks709  tasks710  tasks711  tasks712  tasks713  tasks714  tasks715  tasks716  tasks717  tasks718  tasks719  tasks720  tasks721  tasks722  tasks723  tasks724  tasks725  tasks726  tasks727  tasks728  tasks729  tasks730  tasks731  tasks732  tasks733  tasks734  tasks735  tasks736  tasks737  tasks738  tasks739  tasks740  tasks741  tasks742  tasks743  tasks744  tasks745  tasks746  tasks747  tasks748  tasks749  tasks750  tasks751  tasks752  tasks753  tasks754  tasks755  tasks756  tasks757  tasks758  tasks759  tasks760  tasks761  tasks762  tasks763  tasks764  tasks765  tasks766  tasks767  tasks768  tasks769  tasks770  tasks771  tasks772  tasks773  tasks774  tasks775  tasks776  tasks777  tasks778  tasks779  tasks780  tasks781  tasks782  tasks783  tasks784  tasks785  tasks786  tasks787  tasks788  tasks789  tasks790  tasks791  tasks792  tasks793  tasks794  tasks795  tasks796  tasks797  tasks798  tasks799  tasks800  tasks801  tasks802  tasks803  tasks804  tasks805  tasks806  tasks807  tasks808  tasks809  tasks810  tasks811  tasks812  tasks813  tasks814  tasks815  tasks816  tasks817  tasks818  tasks819  tasks820  tasks821  tasks822  tasks823  tasks824  tasks825  tasks826  tasks827  tasks828  tasks829  tasks830  tasks831  tasks832  tasks833  tasks834  tasks835  tasks836  tasks837  tasks838  tasks839  tasks840  tasks841  tasks842  tasks843  tasks844  tasks845  tasks846  tasks847  tasks848  tasks849  tasks850  tasks851  tasks852  tasks853  tasks854  tasks855  tasks856  tasks857  tasks858  tasks859  tasks860  tasks861  tasks862  tasks863  tasks864  tasks865  tasks866  tasks867  tasks868  tasks869  tasks870  tasks871  tasks872  tasks873  tasks874  tasks875  tasks876  tasks877  tasks878  tasks879  tasks880  tasks881  tasks882  tasks883  tasks884  tasks885  tasks886  tasks887  tasks888  tasks889  tasks890  tasks891  tasks892  tasks893  tasks894  tasks895  tasks896  tasks897  tasks898  tasks899  tasks900  tasks901  tasks902  tasks903  tasks904  tasks905  tasks906  tasks907  tasks908  tasks909  tasks910  tasks911  tasks912  tasks913  tasks914  tasks915  tasks916  tasks917  tasks918  tasks919  tasks920  tasks921  tasks922  tasks923  tasks924  tasks925  tasks926  tasks927  tasks928  tasks929  tasks930  tasks931  tasks932  tasks933  tasks934  tasks935  tasks936  tasks937  tasks938  tasks939  tasks940  tasks941  tasks942  tasks943  tasks944  tasks945  tasks946  tasks947  tasks948  tasks949  tasks950  tasks951  tasks952  tasks953  tasks954  tasks955  tasks956  tasks957  tasks958  tasks959  tasks960  tasks961  tasks962  tasks963  tasks964  tasks965  tasks966  tasks967  tasks968  tasks969  tasks970  tasks971  tasks972  tasks973  tasks974  tasks975  tasks976  tasks977  tasks978  tasks979  tasks980  tasks981  tasks982  tasks983  tasks984  tasks985  tasks986  tasks987  tasks988  tasks989  tasks990  tasks991  tasks992  tasks993  tasks994  tasks995  tasks996  tasks997  tasks998  tasks999  tasks1000  tasks1001  tasks1002  tasks1003  tasks1004  tasks1005  tasks1006  tasks1007  tasks1008  tasks1009  tasks1010  tasks1011  tasks1012  tasks1013  tasks1014  tasks1015  tasks1016  tasks1017  tasks1018  tasks1019  tasks1020  tasks1021  tasks1022  tasks1023  tasks1024  tasks1025  tasks1026  tasks1027  tasks1028  tasks1029  tasks1030  tasks1031  tasks1032  tasks1033  tasks1034  tasks1035  tasks1036  tasks1037  tasks1038  tasks1039  tasks1040  tasks1041  tasks1042  tasks1043  tasks1044  tasks1045  tasks1046  tasks1047  tasks1048  tasks1049  tasks1050  tasks1051  tasks1052  tasks1053  tasks1054  tasks1055  tasks1056  tasks1057  tasks1058  tasks1059  tasks1060  tasks1061  tasks1062  tasks1063  tasks1064  tasks1065  tasks1066  tasks1067  tasks1068  tasks1069  tasks1070  tasks1071  tasks1072  tasks1073  tasks1074  tasks1075  tasks1076  tasks1077  tasks1078  tasks1079  tasks1080  tasks1081  tasks1082  tasks1083  tasks1084  tasks1085  tasks1086  tasks1087  tasks1088  tasks1089  tasks1090  tasks1091  tasks1092  tasks1093  tasks1094  tasks1095  tasks1096  tasks1097  tasks1098  tasks1099  tasks1100  tasks1101  tasks1102  tasks1103  tasks1104  tasks1105  tasks1106  tasks1107  tasks1108  tasks1109  tasks1110  tasks1111  tasks1112  tasks1113  tasks1114  tasks1115  tasks1116  tasks1117  tasks1118  tasks1119  tasks1120  tasks1121  tasks1122  tasks1123  tasks1124  tasks1125  tasks1126  tasks1127  tasks1128  tasks1129  tasks1130  tasks1131  tasks1132  tasks1133  tasks1134  tasks1135  tasks1136  tasks1137  tasks1138  tasks1139  tasks1140  tasks1141  tasks1142  tasks1143  tasks1144  tasks1145  tasks1146  tasks1147  tasks1148  tasks1149  tasks1150  tasks1151  tasks1152  tasks1153  tasks1154  tasks1155  tasks1156  tasks1157  tasks1158  tasks1159  tasks1160  tasks1161  tasks1162  tasks1163  tasks1164  tasks1165  tasks1166  tasks1167  tasks1168  tasks1169  tasks1170  tasks1171  tasks1172  tasks1173  tasks1174  tasks1175  tasks1176  tasks1177  tasks1178  tasks1179  tasks1180  tasks1181  tasks1182  tasks1183  tasks1184  tasks1185  tasks1186  tasks1187  tasks1188  tasks1189  tasks1190  tasks1191  tasks1192  tasks1193  tasks1194  tasks1195  tasks1196  tasks1197  tasks1198  tasks1199  tasks1200  tasks1201  tasks1202  tasks1203  tasks1204  tasks1205  tasks1206  tasks1207  tasks1208  tasks1209  tasks1210  tasks1211  tasks1212  tasks1213  tasks1214  tasks1215  tasks1216  tasks1217  tasks1218  tasks1219  tasks1220  tasks1221  tasks1222  tasks1223  tasks1224  tasks1225  tasks1226  tasks1227  tasks1228  tasks1229  tasks1230  tasks1231  tasks1232  tasks1233  tasks1234  tasks1235  tasks1236  tasks1237  tasks1238  tasks1239  tasks1240  tasks1241  tasks1242  tasks1243  tasks1244  tasks1245  tasks1246  tasks1247  tasks1248  tasks1249  tasks1250  tasks1251  tasks1252  tasks1253  tasks1254  tasks1255  tasks1256  tasks1257  tasks1258  tasks1259  tasks1260  tasks1261  tasks1262  tasks1263  tasks1264  tasks1265  tasks1266  tasks1267  tasks1268  tasks1269  tasks1270  tasks1271  tasks1272  tasks1273  tasks1274  tasks1275  tasks1276  tasks1277  tasks1278  tasks1279  tasks1280  tasks1281  tasks1282  tasks1283  tasks1284  tasks1285  tasks1286  tasks1287  tasks1288  tasks1289  tasks1290  tasks1291  tasks1292  tasks1293  tasks1294  tasks1295  tasks1296  tasks1297  tasks1298  tasks1299  tasks1300  tasks1301  tasks1302  tasks1303  tasks1304  tasks1305  tasks1306  tasks1307  tasks1308  tasks1309  tasks1310  tasks1311  tasks1312  tasks1313  tasks1314  tasks1315  tasks1316  tasks1317  tasks1318  tasks1319  tasks1320  tasks1321  tasks1322  tasks1323  tasks1324  tasks1325  tasks1326  tasks1327  tasks1328  tasks1329  tasks1330  tasks1331  tasks1332  tasks1333  tasks1334  tasks1335  tasks1336  tasks1337  tasks1338  tasks1339  tasks1340  tasks1341  tasks1342  tasks1343  tasks1344  tasks1345  tasks1346  tasks1347  tasks1348  tasks1349  tasks1350  tasks1351  tasks1352  tasks1353  tasks1354  tasks1355  tasks1356  tasks1357  tasks1358  tasks1359  tasks1360  tasks1361  tasks1362  tasks1363  tasks1364  tasks1365  tasks1366  tasks1367  tasks1368  tasks1369  tasks1370  tasks1371  tasks1372  tasks1373  tasks1374  tasks1375  tasks1376  tasks1377  tasks1378  tasks1379  tasks1380  tasks1381  tasks1382  tasks1383  tasks1384  tasks1385  tasks1386  tasks1387  tasks1388  tasks1389  tasks1390  tasks1391  tasks1392  tasks1393  tasks1394  tasks1395  tasks1396  tasks1397  tasks1398  tasks1399  tasks1400  tasks1401  tasks1402  tasks1403  tasks1404  tasks1405  tasks1406  tasks1407  tasks1408  tasks1409  tasks1410  tasks1411  tasks1412  tasks1413  tasks1414  tasks1415  tasks1416  tasks1417  tasks1418  tasks1419  tasks1420  tasks1421  tasks1422  tasks1423  tasks1424  tasks1425  tasks1426  tasks1427  tasks1428  tasks1429  tasks1430  tasks1431  tasks1432  tasks1433  tasks1434  tasks1435  tasks1436  tasks1437  tasks1438  tasks1439  tasks1440  tasks1441  tasks1442  tasks1443  tasks1444  tasks1445  tasks1446  tasks1447  tasks1448  tasks1449  tasks1450  tasks1451  tasks1452  tasks1453  tasks1454  tasks1455  tasks1456  tasks1457  tasks1458  tasks1459  tasks1460  tasks1461  tasks1462  tasks1463  tasks1464  tasks1465  tasks1466  tasks1467  tasks1468  tasks1469  tasks1470  tasks1471  tasks1472  tasks1473  tasks1474  tasks1475  tasks1476  tasks1477  tasks1478  tasks1479  tasks1480  tasks1481  tasks1482  tasks1483  tasks1484  tasks1485  tasks1486  tasks1487  tasks1488  tasks1489  tasks1490  tasks1491  tasks1492  tasks1493  tasks1494  tasks1495  tasks1496  tasks1497  tasks1498  tasks1499  tasks1500  tasks1501  tasks1502  tasks1503  tasks1504  tasks1505  tasks1506  tasks1507  tasks1508  tasks1509  tasks1510  tasks1511  tasks1512  tasks1513  tasks1514  tasks1515  tasks1516  tasks1517  tasks1518  tasks1519  tasks1520  tasks1521  tasks1522  tasks1523  tasks1524  tasks1525  tasks1526  tasks1527  tasks1528  tasks1529  tasks1530  tasks1531  tasks1532  tasks1533  tasks1534  tasks1535  tasks1536  tasks1537  tasks1538  tasks1539  tasks1540  tasks1541  tasks1542  tasks1543  tasks1544  tasks1545  tasks1546  tasks1547  tasks1548  tasks1549  tasks1550  tasks1551  tasks1552  tasks1553  tasks1554  tasks1555  tasks1556  tasks1557  tasks1558  tasks1559  tasks1560  tasks1561  tasks1562  tasks1563  tasks1564  tasks1565  tasks1566  tasks1567  tasks1568  tasks1569  tasks1570  tasks1571  tasks1572  tasks1573  tasks1574  tasks1575  tasks1576  tasks1577  tasks1578  tasks1579  tasks1580  tasks1581  tasks1582  tasks1583  tasks1584  tasks1585  tasks1586  tasks1587  tasks1588  tasks1589  tasks1590  tasks1591  tasks1592  tasks1593  tasks1594  tasks1595  tasks1596  tasks1597  tasks1598  tasks1599  tasks1600  tasks1601  tasks1602  tasks1603  tasks1604  tasks1605  tasks1606  tasks1607  tasks1608  tasks1609  tasks1610  tasks1611  tasks1612  tasks1613  tasks1614  tasks1615  tasks1616  tasks1617  tasks1618  tasks1619  tasks1620  tasks1621  tasks1622  tasks1623  tasks1624  tasks1625  tasks1626  tasks1627  tasks1628  tasks1629  tasks1630  tasks1631  tasks1632  tasks1633  tasks1634  tasks1635  tasks1636  tasks1637  tasks1638  tasks1639  tasks1640  tasks1641  tasks1642  tasks1643  tasks1644  tasks1645  tasks1646  tasks1647  tasks1648  tasks1649  tasks1650  tasks1651  tasks1652  tasks1653  tasks1654  tasks1655  tasks1656  tasks1657  tasks1658  tasks1659  tasks1660  tasks1661  tasks1662  tasks1663  tasks1664  tasks1665  tasks1666  tasks1667  tasks1668  tasks1669  tasks1670  tasks1671  tasks1672  tasks1673  tasks1674  tasks1675  tasks1676  tasks1677  tasks1678  tasks1679  tasks1680  tasks1681  tasks1682  tasks1683  tasks1684  tasks1685  tasks1686  tasks1687  tasks1688  tasks1689  tasks1690  tasks1691  tasks1692  tasks1693  tasks1694  tasks1695  tasks1696  tasks1697  tasks1698  tasks1699  tasks1700  tasks1701  tasks1702  tasks1703  tasks1704  tasks1705  tasks1706  tasks1707  tasks1708  tasks1709  tasks1710  tasks1711  tasks1712  tasks1713  tasks1714  tasks1715  tasks1716  tasks1717  tasks1718  tasks1719  tasks1720  tasks1721  tasks1722  tasks1723  tasks1724  tasks1725  tasks1726  tasks1727  tasks1728  tasks1729  tasks1730  tasks1731  tasks1732  tasks1733  tasks1734  tasks1735  tasks1736  tasks1737  tasks1738  tasks1739  tasks1740  tasks1741  tasks1742  tasks1743  tasks1744  tasks1745  tasks1746  tasks1747  tasks1748  tasks1749  tasks1750  tasks1751  tasks1752  tasks1753  tasks1754  tasks1755  tasks1756  tasks1757  tasks1758  tasks1759  tasks1760  tasks1761  tasks1762  tasks1763  tasks1764  tasks1765  tasks1766  tasks1767  tasks1768  tasks1769  tasks1770  tasks1771  tasks1772  tasks1773  tasks1774  tasks1775  tasks1776  tasks1777  tasks1778  tasks1779  tasks1780  tasks1781  tasks1782  tasks1783  tasks1784  tasks1785  tasks1786  tasks1787  tasks1788  tasks1789  tasks1790  tasks1791  tasks1792  tasks1793  tasks1794  tasks1795  tasks1796  tasks1797  tasks1798  tasks1799  tasks1800  tasks1801  tasks1802  tasks1803  tasks1804  tasks1805  tasks1806  tasks1807  tasks1808  tasks1809  tasks1810  tasks1811  tasks1812  tasks1813  tasks1814  tasks1815  tasks1816  tasks1817  tasks1818  tasks1819  tasks1820  tasks1821  tasks1822  tasks1823  tasks1824  tasks1825  tasks1826  tasks1827  tasks1828  tasks1829  tasks1830  tasks1831  tasks1832  tasks1833  tasks1834  tasks1835  tasks1836  tasks1837  tasks1838  tasks1839  tasks1840  tasks1841  tasks1842  tasks1843  tasks1844  tasks1845  tasks1846  tasks1847  tasks1848  tasks1849  tasks1850  tasks1851  tasks1852  tasks1853  tasks1854  tasks1855  tasks1856  tasks1857  tasks1858  tasks1859  tasks1860  tasks1861  tasks1862  tasks1863  tasks1864  tasks1865  tasks1866  tasks1867  tasks1868  tasks1869  tasks1870  tasks1871  tasks1872  tasks1873  tasks1874  tasks1875  tasks1876  tasks1877  tasks1878  tasks1879  tasks1880  tasks1881  tasks1882  tasks1883  tasks1884  tasks1885  tasks1886  tasks1887  tasks1888  tasks1889  tasks1890  tasks1891  tasks1892  tasks1893  tasks1894  tasks1895  tasks1896  tasks1897  tasks1898  tasks1899  tasks1900  tasks1901  tasks1902  tasks1903  tasks1904  tasks1905  tasks1906  tasks1907  tasks1908  tasks1909  tasks1910  tasks1911  tasks1912  tasks1913  tasks1914  tasks1915  tasks1916  tasks1917  tasks1918  tasks1919  tasks1920  tasks1921  tasks1922  tasks1923  tasks1924  tasks1925  tasks1926  tasks1927  tasks1928  tasks1929  tasks1930  tasks1931  tasks1932  tasks1933  tasks1934  tasks1935  tasks1936  tasks1937  tasks1938  tasks1939  tasks1940  tasks1941  tasks1942  tasks1943  tasks1944  tasks1945  tasks1946  tasks1947  tasks1948  tasks1949  tasks1950  tasks1951  tasks1952  tasks1953  tasks1954  tasks1955  tasks1956  tasks1957  tasks1958  tasks1959  tasks1960  tasks1961  tasks1962  tasks1963  tasks1964  tasks1965  tasks1966  tasks1967  tasks1968  tasks1969  tasks1970  tasks1971  tasks1972  tasks1973  tasks1974  tasks1975  tasks1976  tasks1977  tasks1978  tasks1979  tasks1980  tasks1981  tasks1982  tasks1983  tasks1984  tasks1985  tasks1986  tasks1987  tasks1988  tasks1989  tasks1990  tasks1991  tasks1992  tasks1993  tasks1994  tasks1995  tasks1996  tasks1997  tasks1998  tasks1999  tasks2000  tasks2001  tasks2002  tasks2003  tasks2004  tasks2005  tasks2006  tasks2007  tasks2008  tasks2009  tasks2010  tasks2011  tasks2012  tasks2013  tasks2014  tasks2015  tasks2016  tasks2017  tasks2018  tasks2019  tasks2020  tasks2021  tasks2022  tasks2023  tasks2024  tasks2025  tasks2026  tasks2027  tasks2028  tasks2029  tasks2030  tasks2031  tasks2032  tasks2033  tasks2034  tasks2035  tasks2036  tasks2037  tasks2038  tasks2039  tasks2040  tasks2041  tasks2042  tasks2043  tasks2044  tasks2045  tasks2046  tasks2047  tasks2048  tasks2049  tasks2050  tasks2051  tasks2052  tasks2053  tasks2054  tasks2055  tasks2056  tasks2057  tasks2058  tasks2059  tasks2060  tasks2061  tasks2062  tasks2063  tasks2064  tasks2065  tasks2066  tasks2067  tasks2068  tasks2069  tasks2070  tasks2071  tasks2072  tasks2073  tasks2074  tasks2075  tasks2076  tasks2077  tasks2078  tasks2079  tasks2080  tasks2081  tasks2082  tasks2083  tasks2084  tasks2085  tasks2086  tasks2087  tasks2088  tasks2089  tasks2090  tasks2091  tasks2092  tasks2093  tasks2094  tasks2095  tasks2096  tasks2097  tasks2098  tasks2099  tasks2100  tasks2101  tasks2102  tasks2103  tasks2104  tasks2105  tasks2106  tasks2107  tasks2108  tasks2109  tasks2110  tasks2111  tasks2112  tasks2113  tasks2
```

```

1. Create file
2. List files
3. View file
4. Delete file
5. Edit file
6. Exit
Choose an option: 2
Files:
- hello.c (33 bytes, 1 blocks used)
- bigfile.sh (1023 bytes, 4 blocks used)

```

Fig.9. Comparison of Files

Since the content of "bigfile.sh" is larger than "hello.c," the details in the list view are updated accordingly. It shows that "bigfile.sh" contains 1 KB of data and uses 4 blocks of storage.

```

1. Create file
2. List files
3. View file
4. Delete file
5. Edit file
6. Exit
Choose an option: 3
Enter file name to view: bigfile.sh
Content of 'bigfile.sh':
askdmasaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaa

```

Fig. 10 View File

The View File feature allows users to view the contents of a file without the ability to edit its current content. This feature is particularly useful for accessing and tracking files, especially when multiple files are created, providing users with a read-only preview for easy reference.

```

1. Create file
2. List files
3. View file
4. Delete file
5. Edit file
6. Exit
Choose an option: 4
Enter file name to delete: bigfile.sh
File 'bigfile.sh' deleted.

1. Create file
2. List files
3. View file
4. Delete file
5. Edit file
6. Exit
Choose an option: 2
Files:
- hello.c (33 bytes, 1 blocks used)

```

Fig. 11. Delete File

The Delete File feature allows users to remove a file by entering its name. After attempting to delete the file, the user will be notified whether the deletion was successful. To verify if the file has been deleted, the user can select option 2 to view the list of files. If the file was successfully deleted, it will no longer appear in the directory.

```

1. Create file
2. List files
3. View file
4. Delete file
5. Edit file
6. Exit
Choose an option: 5
Enter file name to edit: hello.c
Enter new content: this is my new content. Hello world!
File 'hello.c' updated successfully.

1. Create file
2. List files
3. View file
4. Delete file
5. Edit file
6. Exit
Choose an option: 3
Enter file name to view: hello.c
Content of 'hello.c':
this is my new content. Hello world!

```

Fig. 12. Edit File

The Edit File feature allows users to modify the content of an existing file if there are any mistakes. This is more efficient than creating a new file to replace the incorrect content, as it helps conserve space on the virtual disk.

#### A. Code Used:

file\_system.sh:

```

#!/bin/bash

VIRTUAL_DISK="virtual_disk.img"
DISK_SIZE=1048576 # 1MB virtual disk

# Initialize Virtual Disk
init_disk() {
    if [ ! -f $VIRTUAL_DISK ]; then
        dd if=/dev/zero of=$VIRTUAL_DISK bs=1
        count=$DISK_SIZE
        echo "Virtual disk initialized: $VIRTUAL_DISK"
    else
        echo "Virtual disk already exists."
    fi
}

# Run File System Operations
run_fs() {
    gcc file_system.c -o file_system
    ./file_system
}

# Main Menu
echo "Custom File System"
echo "1. Initialize Virtual Disk"
echo "2. Run File System"
echo "3. Exit"
read -p "Choose an option: " option

case $option in
    1) init_disk ;;
    2) run_fs ;;
    3) exit ;;
    *) echo "Invalid option";;
esac

```

file\_system.c:

```

#include <stdio.h>
#include <stdlib.h>

```

```

#include <string.h>
#include <stdbool.h>

#define BLOCK_SIZE 256
#define MAX_BLOCKS 4096
#define MAX_FILES 128

// Allocation type
#define ALLOC_CONTIGUOUS 0

// File entry structure
typedef struct {
    char name[32];
    int start_block;
    int size_bytes; // actual content length
    int blocks_used;
    int allocation_type;
    bool used;
} FileEntry;

// File system structure
typedef struct {
    FileEntry files[MAX_FILES];
    bool block_map[MAX_BLOCKS];
} FileSystem;

FileSystem fs;
FILE *virtual_disk;

void init_file_system() {
    memset(&fs, 0, sizeof(fs));
    virtual_disk = fopen("virtual_disk.img", "r+b");
    if (!virtual_disk) {
        virtual_disk = fopen("virtual_disk.img", "w+b");
        fseek(virtual_disk, BLOCK_SIZE *
MAX_BLOCKS - 1, SEEK_SET);
        fputc('\0', virtual_disk);
        fflush(virtual_disk);
    }
}

void shutdown_file_system() {
    fclose(virtual_disk);
}

int find_free_blocks(int blocks_needed) {
    int count = 0;
    for (int i = 0; i <= MAX_BLOCKS -
blocks_needed; i++) {
        bool found = true;
        for (int j = 0; j < blocks_needed; j++) {
            if (fs.block_map[i + j]) {
                found = false;
                break;
            }
        }
        if (found) return i;
    }
    return -1;
}

void create_file(const char *name, const char *content) {
    for (int i = 0; i < MAX_FILES; i++) {
        if (fs.files[i].used && strcmp(fs.files[i].name,
name) == 0) {
            printf("File with this name already exists.\n");
            return;
        }
    }
}

```

```

        int content_length = strlen(content);
        int blocks_needed = (content_length +
BLOCK_SIZE - 1) / BLOCK_SIZE;

        int start_block =
find_free_blocks(blocks_needed);
        if (start_block == -1) {
            printf("Not enough contiguous space
available.\n");
            return;
        }

        for (int i = start_block; i < start_block +
blocks_needed; i++) {
            fs.block_map[i] = true;
        }

        for (int i = 0; i < MAX_FILES; i++) {
            if (!fs.files[i].used) {
                strcpy(fs.files[i].name, name);
                fs.files[i].start_block = start_block;
                fs.files[i].size_bytes = content_length;
                fs.files[i].blocks_used = blocks_needed;
                fs.files[i].allocation_type =
ALLOC_CONTIGUOUS;
                fs.files[i].used = true;
                break;
            }
        }

        fseek(virtual_disk, start_block * BLOCK_SIZE,
SEEK_SET);
        fwrite(content, 1, content_length, virtual_disk);
        fflush(virtual_disk);

        printf("File '%s' created successfully.\n", name);
    }

void list_files() {
    printf("Files:\n");
    for (int i = 0; i < MAX_FILES; i++) {
        if (fs.files[i].used) {
            printf("- %s (%d bytes, %d blocks used)\n",
fs.files[i].name,
fs.files[i].size_bytes,
fs.files[i].blocks_used);
        }
    }
}

void view_file(const char *name) {
    for (int i = 0; i < MAX_FILES; i++) {
        if (fs.files[i].used && strcmp(fs.files[i].name,
name) == 0) {
            char *buffer = malloc(fs.files[i].size_bytes + 1);
            fseek(virtual_disk, fs.files[i].start_block *
BLOCK_SIZE, SEEK_SET);
            fread(buffer, 1, fs.files[i].size_bytes,
virtual_disk);
            buffer[fs.files[i].size_bytes] = '\0';
            printf("Content of '%s':\n%s\n", name, buffer);
            free(buffer);
            return;
        }
    }
    printf("File not found.\n");
}

```

```

void delete_file(const char *name) {
    for (int i = 0; i < MAX_FILES; i++) {
        if (fs.files[i].used && strcmp(fs.files[i].name,
name) == 0) {
            for (int j = fs.files[i].start_block;
j < fs.files[i].start_block +
fs.files[i].blocks_used; j++) {
                fs.block_map[j] = false;
            }
            fs.files[i].used = false;
            printf("File '%s' deleted.\n", name);
            return;
        }
    }
    printf("File not found.\n");
}

void edit_file(const char *name, const char *new_content) {
    for (int i = 0; i < MAX_FILES; i++) {
        if (fs.files[i].used &&
strcmp(fs.files[i].name, name) == 0) {
            int new_length =
strlen(new_content);
            int new_blocks =
(new_length + BLOCK_SIZE - 1) / BLOCK_SIZE;

            // Free old blocks
            for (int j =
fs.files[i].start_block;
j <
fs.files[i].start_block + fs.files[i].blocks_used; j++) {
                fs.block_map[j] = false;
            }

            // Find new space
            int new_start_block =
find_free_blocks(new_blocks);
            if (new_start_block == -1)
            {
                printf("Not
enough contiguous space available for updated content.\n");

                // Reallocate
                the old blocks back if new space can't be found
                for (int j =
fs.files[i].start_block;
j <
fs.files[i].start_block + fs.files[i].blocks_used; j++) {
                    fs.block_map[j] = true;
                }
            }
            return;
        }
    }

    for (int j =
new_start_block; j < new_start_block + new_blocks; j++) {
        fs.block_map[j] = true;
    }

    fs.files[i].start_block =
new_start_block;
    fs.files[i].size_bytes =
new_length;
    fs.files[i].blocks_used =
new_blocks;

```

```

        fseek(virtual_disk,
new_start_block * BLOCK_SIZE, SEEK_SET);
        fwrite(new_content, 1,
new_length, virtual_disk);
        fflush(virtual_disk);

        printf("File '%s' updated
successfully.\n", name);
        return;
    }
    printf("File not found.\n");
}

int main() {
    init_file_system();

    int choice;
    char name[32];
    char content[1024];

    while (1) {
        printf("\n1. Create file\n2. List files\n3. View
file\n4. Delete file\n5. Edit file\n6. Exit\nChoose an option:
");
        scanf("%d", &choice);
        getchar();

        switch (choice) {
            case 1:
                printf("Enter file name: ");
                fgets(name, sizeof(name), stdin);
                name[strcspn(name, "\n")] = '\0';
                printf("Enter file content: ");
                fgets(content, sizeof(content), stdin);
                content[strcspn(content, "\n")] = '\0';
                create_file(name, content);
                break;
            case 2:
                list_files();
                break;
            case 3:
                printf("Enter file name to view: ");
                fgets(name, sizeof(name), stdin);
                name[strcspn(name, "\n")] = '\0';
                view_file(name);
                break;
            case 4:
                printf("Enter file name to delete: ");
                fgets(name, sizeof(name), stdin);
                name[strcspn(name, "\n")] = '\0';
                delete_file(name);
                break;
            case 5:
                printf("Enter file name to edit: ");
                fgets(name, sizeof(name), stdin);
                name[strcspn(name, "\n")] = '\0';
                printf("Enter new content: ");
                fgets(content, sizeof(content), stdin);
                content[strcspn(content, "\n")] = '\0';
                edit_file(name, content);
                break;
            case 6:
                shutdown_file_system();
                return 0;
            default:

```

```

        printf("Invalid option.\n");
    }
}

```

### X1. ANALYSIS OF THE DATA

The CORE file system emulator, which was implemented and tested in a Fedora Linux environment, successfully demonstrates file system operations through contiguous allocation. The system allows users to initialize a virtual disk, open files, read and write content, and control file deletions. The virtual disk is established in the directory during initialization, and the ls command confirms its characteristics. It has variable storage capacities that can be modified as required. Users can define files by name and content, and the system monitors file size and utilization of blocks, as observed with large files such as "bigfile.sh," which occupies more than one block for storage. The List Files function enables users to see all the files with their size and block utilization, while the View File option offers a read-only view of file content. The Delete File feature ensures that files can be removed, with confirmation of successful deletion provided to the user, and the Edit File feature allows users to modify file content without creating new files, conserving disk space. The emulator showcases effective file management and provides a simple interface for tracking and managing files in a virtual disk system.

### XIII. CONCLUSION

CORE: Custom OS-level Resource Emulator provides a crucial educational tool that bridges theoretical knowledge and practical experience in operating system design. By simulating file allocation processes using contiguous block allocation, CORE allows students to visualize the internal workings of file systems. This hands-on approach supports the growing demand for interactive learning platforms that enhance understanding of complex systems, as demonstrated by Ferreira et al. with their web-based oscilloscope interface [27]. CORE helps demystify OS-level operations, offering a practical, user-friendly environment. Furthermore, inspired by Alzakari et al.'s work on using deep learning for security in IoT environments [28], there is potential for CORE to address real-world issues such as security in file systems. Future developments could incorporate features like ransomware detection, offering students both system management and security insights. CORE's ability to simulate and interact with file systems, while also addressing potential security vulnerabilities, significantly enhances its educational value. Overall, CORE not only fills a gap in systems programming education by making abstract concepts tangible but also opens opportunities for advancements in OS education. Expanding CORE's functionality could allow students to explore a wider range of OS operations and security challenges, preparing them for both academic and industry problems in computer engineering.

### XII. RECOMMENDATION

To further enhance the CORE: Custom OS-level Resource Emulator, the integration of blended learning strategies, as outlined by Amarathunga et al. [29], could significantly optimize the learning process. By combining asynchronous online modules, interactive learning environments, and real-time simulations, students would have the opportunity to practice system-level simulations remotely, supporting self-regulated learning and fostering collaborative problem-solving. Additionally, the application of simulation-based learning, as demonstrated by Pakosch et al. in their "StudyTalk" project, could offer students the opportunity to manage and design

virtual file systems in a sandboxed environment. This would allow learners to engage in realistic system configurations and error-handling tasks, mimicking actual industry scenarios [30]. Incorporating techniques that address motivation, self-efficacy, and collaborative learning, as suggested by Amarathunga et al., could also enhance the effectiveness of CORE, enabling students to improve their practical skills while engaging with complex systems programming challenges [29]. By embedding project-based, real-time system management simulations, CORE can move beyond theoretical OS design, offering a robust platform for experiential learning, thus preparing students for future roles in systems programming, data management, and security.

### REFERENCES

- [1] Jiang, J., Yang, M., Qiao, L., Wang, T., & Chen, X. (2025). MIFS: A low overhead and efficient mixture file index management method in flash file system. *Journal of Systems Architecture*, 162. Scopus. <https://doi.org/10.1016/j.sysarc.2025.103387>
- [2] Khan, A., Litchfield, A., Alabdulatif, A., & Khan, F. (2025). BlockPres IPFS: Performance evaluation of blockchain based secure patients prescription record storage using IPFS for smart prescription management system. *Cluster Computing*, 28(4). Scopus. <https://doi.org/10.1007/s10586-024-05054-6>
- [3] Anand, P., Singh, Y., & Singh, H. (2025). Secure IoT data dissemination with blockchain and transfer learning techniques. *Scientific Reports*, 15(1). Scopus. <https://doi.org/10.1038/s41598-024-84837-8>
- [4] Kim, H., Heo, G., & Doh, I. (2025). Role-based federated learning exploiting IPFS for privacy enhancement in IoT environment. *Computer Networks*, 263. Scopus. <https://doi.org/10.1016/j.comnet.2025.111200>
- [5] Szekely, A., Belay, A., Morris, R., & Kaashoek, M. F. (2024). *Unifying serverless and microservice workloads with SigmaOS*. 385–402. Scopus. <https://doi.org/10.1145/3694715.3695947>
- [6] Gandhi, H. S., & Deshmukh, A. D. (2024). Understanding File System Forensics: FAT, NTFS, HFS, and EXT. In *Digital Forensics in the Age of Ai* (pp. 177–194). Scopus. <https://doi.org/10.4018/979-8-3373-0857-9.ch007>
- [7] Toolan, F. (2025). File system forensics (p. 464). Scopus.
- [8] Marková, E., Sokol, P., Křišáková, S. P., & Kováčová, K. (2024). Dataset of Windows operating system forensics artefacts. *Data in Brief*, 55. Scopus. <https://doi.org/10.1016/j.dib.2024.110693>
- [9] Deak, T., Deaconescu, R., Kroning, M., & Monti, A. (2024). High-Configurability for Operating Systems. *Proceedings - RoEduNet IEEE International Conference*. Scopus. <https://doi.org/10.1109/RoEduNet64292.2024.10722499>
- [10] Hao, D., Gao, C., & Shu, J. (2025). Design of User Space Storage Architecture for SCSI Subsystem. *Jisuanji Yanjiu Yu Fazhan/Computer Research and Development*, 62(3), 633–647. Scopus. <https://doi.org/10.7544/issn1000-1239.202404632>
- [11] Batra, R., Mandal, S. K., Singh, D., Parmar, Y., Bhatt, M., & Kharbas, V. K. (2025). Novel method for preventing loss of information in multi-cloud storage assistance. *International Journal of System Assurance Engineering and Management*. Scopus. <https://doi.org/10.1007/s13198-024-02681-5>
- [12] Farina, D. A. R., Campiolo, R., Rufino, J., & Pereira, M. J. V. (2024). Automatic and Dynamic Visualization of Process-Based Concurrent Programs. 120. Scopus. <https://doi.org/10.4230/OASlcs.SLATE.2024.6>
- [13] Kegeleirs, M., Garzón Ramos, D., & Birattari, M. (2025). DeimOS: A ROS-Ready operating system for the e-puck. *Journal of Open Research Software*, 13(1). Scopus. <https://doi.org/10.5334/jors.437>
- [14] Mahajan, J., & Prachi, A. (2024). Decentralized File Storage: Leveraging Blockchain, Polygon, Web3, and IPFS. 2024 Parul International Conference on Engineering and Technology, PICET 2024. Scopus. <https://doi.org/10.1109/PICET60765.2024.10716189>
- [15] Yu, Z., Zhang, R., Yang, C., Nie, S., & Liu, D. (2023). An efficient wear-leveling-aware multi-grained allocator for persistent memory file systems. *Frontiers of Information Technology and Electronic Engineering*, 24(5), 688–702. Scopus. <https://doi.org/10.1631/FITEE.2200468>
- [16] Liu, J.-K., & Wang, S.-D. (2022). CFFS: A Persistent Memory File System for Contiguous File Allocation With Fine-Grained Metadata. *IEEE Access*, 10, 91678–91698. Scopus. <https://doi.org/10.1109/ACCESS.2022.3202532>
- [17] Mune Gowda, K., Raju, G. T., Raju, V. M., & Manjunath, T. N. (2015). Characterization of transaction-safe cluster allocation strategies of textat file system for embedded storage devices. *Smart Innovation, Systems and Technologies*, 31, 503–515. Scopus. [https://doi.org/10.1007/978-81-322-2205-7\\_47](https://doi.org/10.1007/978-81-322-2205-7_47)



- [18] Devulapalli, A., Dalessandro, D., Wyckoff, P., Ali, N., & Sadayappan, P. (2007). Integrating parallel file systems with object-based storage devices. *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, SC'07*. Scopus. <https://doi.org/10.1145/1362622.1362659>
- [19] Peng, H.-D., Chen, Y.-S., Chen, T.-Y., & Chang, Y.-H. (2025). PULSE: Progressive Utilization of Log-Structured Techniques to Ease SSD Write Amplification in B-epsilon-tree. 245–250. Scopus. <https://doi.org/10.1145/3658617.3697561>
- [20] Mishra, D. P., Rajeev, B., Mallick, S. R., Lenka, R. K., & Salkuti, S. R. (2025). Efficient blockchain based solution for secure medical record management. *International Journal of Informatics and Communication Technology*, 14(1), 59–67. Scopus. <https://doi.org/10.11591/ijict.v14i1.pp59-67>
- [21] Yang, Y., Liu, M., Wang, L., Pu, Y., Zheng, R., Wu, Q., & Guo, J. (2025). A ring signature scheme with linkability and traceability for blockchain-based medical data sharing system. *Peer-to-Peer Networking and Applications*, 18(3). Scopus. <https://doi.org/10.1007/s12083-025-01913-0>
- [22] Mythili, J., & Gopalakrishnan, R. (2025). Improving data transmission through optimizing blockchain sharding in cloud IoT based healthcare applications. *Egyptian Informatics Journal*, 30. Scopus. <https://doi.org/10.1016/j.eij.2025.100661>
- [23] Nguyen, L.-D., & Leis, V. (2024). Why Files if You Have a DBMS? 3878–3892. Scopus. <https://doi.org/10.1109/ICDE60146.2024.00297>
- [24] Duong, T. D., & Hur, J. Y. (2025). Contiguity aware TLB prefetching for embedded I/O devices. *IEICE Electronics Express*, 22(3). Scopus. <https://doi.org/10.1587/elex.21.20240664>
- [25] Samy, A., Elgendy, I. A., Yu, H., Zhang, W., & Zhang, H. (2022). Secure Task Offloading in Blockchain-Enabled Mobile Edge Computing With Deep Reinforcement Learning. *IEEE Transactions on Network and Service Management*, 19(4), 4872–4887. Scopus. <https://doi.org/10.1109/TNSM.2022.3190493>
- [26] Lv, F. (2025). Research on optimization strategies of university ideological and political parenting models under the empowerment of digital intelligence. *Scientific Reports*, 15(1). Scopus. <https://doi.org/10.1038/s41598-025-92985-8>
- [27] Ferreira, J., Rocha, A., Alves, M., & Oliveira, P. C. D. (2025). On a Web-Based Oscilloscope Interface App for E-Learning: Software Architecture, Practical Applications, and User Experience. *Sci*, 7(1). Scopus. <https://doi.org/10.3390/sci7010019>
- [28] Alzakari, S. A., Aljebreen, M., Ahmad, N., Alhashmi, A. A., Alahmari, S., Alrusaini, O., Al-Sharafi, A. M., & Almukadi, W. S. (2025). An intelligent ransomware based cyberthreat detection model using multi head attention-based recurrent neural networks with optimization algorithm in IoT environment. *Scientific Reports*, 15(1). Scopus. <https://doi.org/10.1038/s41598-025-92711-4>
- [29] Amarathunga, B. (2025). Blended learning trends and future directions: A systematic literature review and bibliometric analysis. *International Journal of Information and Learning Technology*, 42(2), 147–164. Scopus. <https://doi.org/10.1108/IJILT-04-2024-0072>
- [30] Pakosch, A., Allerkamp, D., & Koschel, A. (2025). Simulating Real IT Project Life for Students. 2, 1569–1570. Scopus. <https://doi.org/10.1145/3641555.3705249>
- [30]