Shaurya chouhan

Extra credit

Read

I've accomplished building JBOD, a storage system with various disks. My coding project was divided into functions, each with its own duties like read data, mount or unmount. Utilizing bitwise operations, I delved into the realm of coding and created codes to communicate instructions effectively to the system. It was an enthralling experience, almost feeling like I possessed a secret language exclusively known to the computer. I implanted mechanisms to handle errors and checked for mistakes beforehand, realizing their significance. My usage of dynamic memory allocation enabled the efficient management of system resources by permitting me to ask for and let go of space as required.

Within the code I wrote, I ensured the implementation of safety measures, which protected the stability and reliability of the storage system and minimized errors and unanticipated behavior, and it was akin to the use of safety nets. Leveraged into the pursuit of function and efficiency, I obtained a library of pre-existing functions. Like borrowing from a proficient friend to hasten task completion, this became a valuable resource that I could take advantage of. The computer's actions in locating and fetching information were efficiently guided by the code that referenced disks, blocks, and data retrieval.

Overall, this was good for growth but was hard for someone with almost no coding background.

write

Throughout my coding journey, I have come to understand the criticality of seeking permission before making any changes to data. This parallels the concept of checking authorization before proceeding with writing operations. Additionally, I have grasped the importance of revoking this permission once the writing task is completed, like the act of closing a door after finishing a specific task.

Before initiating any writing operations, it is essential to conduct a thorough evaluation. This includes verifying permission, confirming the mounted status of the storage, and ensuring the overall coherence of the system. This cautious approach can be likened to checking traffic lights before commencing a drive, emphasizing the significance of a well-prepared and responsible process. By adhering to these practices, I establish a solid foundation for the careful manipulation of data.

Once the necessary checks are complete, I proceed with the exciting task of writing the data. Adopting a systematic approach, I break down the data into cohesive blocks. Each block is meticulously read, updated with new data, and then reintegrated into the system. This methodology mirrors the process of updating a page within a substantial book, maintaining the overall structure and integrity of the data.

My coding journey has provided me with valuable insights into the responsible acquisition and handling of permission within the realm of computer storage. By adhering to protocols and effectively manipulating data, I have cultivated a skill set that enables me to write data accurately and maintain the integrity of the entire system. This experience underscores the importance of meticulousness and adherence to established protocols for optimal data manipulation.

Cache

During my internship, I gained hands-on experience in building and managing a cache system. I started by creating a cache with a specific number of entries, ensuring that the size was within a valid range. This involved allocating space for the cache and initializing it accordingly.

To make the cache fully functional, I implemented three important functions: lookup, update, and insert. The lookup function was responsible for searching the cache for a match and returning the corresponding data if found. It also updated the access time of the entry. The update function allowed for modifying existing cache entries, while the insert function enabled the addition of new entries using the Most Recently Used (MRU) strategy.

To ensure data integrity and proper operation, I incorporated input validation checks in each function. These checks included verifying that the cache was enabled and that the parameters provided were within the appropriate ranges. Additionally, I tracked the hit rate of the cache, which provided valuable insights into the effectiveness of the cache system.

To enhance the flexibility of the cache, I implemented cache resizing functionality. This allowed for adapting the cache size to changing requirements as needed, making the system more versatile and adaptable.

Overall, this coding journey significantly enriched my understanding of cache management. From the initial creation and initialization of the cache to the efficient utilization of different cache functions and even implementing resizing capabilities, I acquired a comprehensive knowledge of managing cache systems effectively.

Networking

During my coding journey, I gained diverse experience in socket programming, particularly in building a client-server communication system for a storage application. The code includes functions such as jbod\_connect and jbod\_disconnect that establish and close connections using sockets, respectively.

To facilitate communication between the client and the server, I implemented send\_packet and recv\_packet functions that handle the transmission and reception of packets, adhering to the specified communication protocol. These functions ensure the reliable exchange of data between the client and the server.

The main orchestrator of the client's operation, jbod\_client\_operation, handles the overall process. This function sends requests to the server, receives responses, and verifies successful execution of the requested operation.

To mitigate potential issues and ensure proper error handling, I incorporated appropriate checks throughout the code. This includes handling scenarios such as invalid IP addresses, failed socket creation, and connection problems. Additionally, functions like nread and nwrite were implemented to enable reliable reading and writing of data to and from sockets.

Through this coding experience, I deepened my understanding of socket programming, client-server communication, and error handling in a networked environment. It also provided valuable insights into building an effective communication protocol for a storage system.

Future enhancements

After learning a lot during my internship, I think we can make the JBOD system even better in the future. We could improve how the system talks and shares information by fixing errors more smartly, making sure data is super secure, and finding faster ways for the system to communicate.

Another important thing we could do is make the system able to handle more stuff. Right now, it works with a certain number of disks, blocks, and users, but we can make it handle even more without slowing down. We might add features that balance the load or manage storage across different parts of the system.

We can also make the JBOD system do more cool things. We might make it better at managing information and handling different types of storage tasks. This would make it useful in more situations.

Making the system use resources better and respond faster is another big goal. This might mean looking at how the system is doing, making the code work even smoother, and using some clever tricks to get things done faster.

In short, the plan for the future is to make the JBOD system work even better by fixing how it talks, handling more stuff, adding cool features, and making it faster and more efficient.