

Working on this project taught me several valuable lessons:

1. **Breaking Down the Problem:**

I learned to divide the overall task (managing a contiguous memory region) into smaller, clear parts:

- Requesting memory (using first fit, best fit, or worst fit)
- Releasing memory
- Compacting memory (merging free blocks)
- Reporting the current memory map

2. **Designing the Data Structure:**

I chose a linked list to represent memory blocks. Each node in the list holds the start address, end address, available space, and process identifier. This structure made insertion, deletion, and merging operations efficient and kept the code modular.

3. **Step-by-Step Implementation:**

Each function was built with a clear purpose. For example, the `create_node` function handles the splitting of a block into an allocated part and a leftover free part. Using helper functions like `merge` and `swap_nodes` allowed me to tackle complex operations (like compaction) in manageable steps.

4. **Consistency and Validation:**

I ensured that the pointers and address values always remained valid by checking boundaries (e.g., not exceeding the highest address). Error checking for user input was also incorporated using functions like `read` and `sscanf`.

5. **Modularity and Reusability:**

By writing separate functions for each operation (requesting, releasing, compacting, reporting), I made the program easier to understand, maintain, and extend. This modular approach is key in both academic assignments and real-world projects.

Overall, this project deepened my understanding of how operating systems manage memory and taught me to apply problem-solving techniques such as decomposition, validating constraints, and modular coding.

Thank you,

Masoud Rafiee