Design and Implementation of a Memory Management Simulator

Harrish Bansal
Enrollment No: 23119012

January 5, 2026

Contents

1 Introduction

Memory management is one of the core responsibilities of an operating system. It involves allocation, deallocation, and efficient utilization of physical memory while minimizing fragmen- tation. This project implements a memory management simulator that models dynamic memory allocation strategies commonly used in operating systems.

The simulator is implemented entirely in user space and focuses on algorithmic correctness rather than low-level hardware interactions. Optional extensions such as cache simulation and virtual memory are discussed but not implemented.

2 Memory Layout and Assumptions

The simulator models physical memory as a contiguous linear address space starting from address 0. The total memory size is configurable at runtime using a command-line interface.

Memory is divided into variable-sized blocks. Each block maintains the following metadata:

- Starting address
- Size of the block
- Allocation status (free or allocated)
- Unique block identifier (for allocated blocks)

Internally, memory blocks are maintained as a sequential list, preserving address order.

3 Allocation Strategies

The simulator implements three dynamic memory allocation strategies:

3.1 First Fit

The First Fit strategy scans the memory blocks sequentially and allocates the first free block that is large enough to satisfy the request.

3.2 Best Fit

The Best Fit strategy searches all free blocks and selects the smallest block that can satisfy the allocation request. This approach attempts to minimize wasted space but may increase fragmentation.

3.3 Worst Fit

The Worst Fit strategy allocates memory from the largest available free block. This approach leaves larger remaining free blocks but may lead to inefficient utilization.

3.4 Block Splitting

When a free block is larger than the requested size, it is split into:

- An allocated block of exact requested size
- A remaining free block

As a result, internal fragmentation is eliminated in this implementation.

4 Deallocation and Coalescing

Memory is deallocated using a block identifier. Upon deallocation:

- The block is marked as free
- Adjacent free blocks are merged (coalesced)

Coalescing reduces external fragmentation by combining contiguous free regions into larger blocks.

5 Statistics and Metrics

The simulator computes and reports the following metrics:

- Total memory
- Used memory
- Free memory
- Memory utilization percentage
- Internal fragmentation (zero by design)
- External fragmentation
- Allocation request count
- Successful and failed allocations

External fragmentation is calculated

as:

$$\text{External Fragmentation} = \frac{\text{Total Free Memory} - \text{Largest Free Block}}{\text{Total Free Memory}} \times 100$$

6 Command-Line Interface

The simulator provides an interactive command-line interface supporting the following commands:

- init <size> – Initialize memory
- set <first_fit | best_fit | worst_fit> – Select allocator
- malloc <size> – Allocate memory
- free <block_id> – Free allocated block
- dump – Display memory layout
- stats – Display memory statistics

This interface enables interactive testing and demonstration of allocation behavior.

7 Buddy Allocation System (Optional)

The buddy allocation system was considered as an optional extension. In this scheme:

- Memory is divided into power-of-two sized blocks
- Allocation involves recursive splitting
- Deallocation merges buddy blocks using XOR-based address calculation This feature was not implemented due to time constraints.

8 Cache Simulation (Optional)

Cache simulation was considered as an optional extension. A potential design would include:

- Multi-level cache hierarchy (L1, L2)
- Configurable cache sizes and block sizes
- FIFO replacement policy

Cache simulation was not implemented in the current version.

9 Virtual Memory Simulation (Optional)

Virtual memory using paging was considered as an optional extension. A complete design would involve:

- Page tables
- Address translation
- Page replacement policies such as FIFO or LRU

Virtual memory and paging were not implemented in the current version.

10 Limitations and Simplifications

The following limitations apply to the simulator:

- Single-process memory model
- No concurrency or multithreading
- No cache or virtual memory simulation
- User-space simulation only

11 Conclusion

This project successfully demonstrates the implementation of dynamic memory allocation strate- gies and fragmentation management in a simulated environment. The simulator provides clear visualization, accurate statistics, and a flexible interface for experimentation. Optional exten- sions were analyzed conceptually but excluded from implementation.