

# Contiguous Memory Management Simulator

## 1. Introduction

This document describes the design and implementation of a **contiguous memory management simulator**. The simulator models how an operating system allocates and frees memory blocks using different allocation strategies, tracks fragmentation, and visualizes memory usage in real time via a terminal-based interface.

The project is intended as an educational and experimental tool to understand memory allocation policies, fragmentation behavior, and compaction.

[illegible]

*Fig1 : Simulator Dashboard*

## 2. Memory Layout and Assumptions

## 2.1 Memory Model

- The simulator models **main memory as a single contiguous region** of fixed size.
- Memory size is specified explicitly by the user using the `init memory <size>` command.
- Memory addresses are simulated as offsets from 0 to `total_memory - 1`.
- No paging or virtual memory abstraction is used.

## 2.2 Block Representation

Memory is divided into **blocks**, each represented by:

- block\_id – unique identifier for allocated blocks
- start – starting address of the block
- size – size of the block
- free – whether the block is free or allocated

Free blocks do not have a valid block ID.

## 2.3 Key Assumptions

- Allocation is **contiguous**: each request must be satisfied by a single free block.
- No alignment constraints are enforced.
- No memory protection or permissions are modeled.
- Internal fragmentation inside allocated blocks is ignored; fragmentation is treated as **external fragmentation**.

---

```
Enter command:
> init memory 1024
```

*Fig 2 : Initializing the memory*

```
Enter command:
> help
Available commands:
  init memory <size>           Initialize memory
  set allocator first_fit      Use First-Fit allocator
  set allocator best_fit       Use Best-Fit allocator
  set allocator worst_fit      Use Worst-Fit allocator
  malloc <size>                Allocate memory
  free <block_id>              Free allocated block
  dump                        Show block IDs and memory layout
  help                        Show this help
  exit | quit                  Exit simulator
  compact                      Compact memory blocks

Press Enter to continue...
```

*Fig 3 : 'help' command can be used to see available commands*

### 3. Allocation Strategy Implementations

The simulator supports multiple allocation strategies. Each strategy implements a common interface and can be swapped at runtime.

### 3.1 First-Fit Allocation

**Definition:**

- Scans memory from the beginning and allocates the first free block large enough to satisfy the request.

### Characteristics:

- Fast allocation
- Tends to leave small unusable holes near the front of memory

### Implementation Notes:

- Iterates sequentially through the block list
- Splits the free block if it is larger than requested

```
Memory Management Simulator
```

---

```
Total      : 1024 bytes  
Used       : 154 bytes  
Free       : 870 bytes  
Allocations: 5 Success / 0 Fail (100.0%)  
Ext Frag   : 42.53 %
```

Memory Map:

[■■■■■□□□□□□□□□□□□□□□■□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□]

Legend: ■ = USED   □ = FREE

---

```
Enter command:  
> dump
```

Block Map:

ID	Start	Size	Status
1	0x0	100	USED
-	0x64	300	FREE
3	0x190	44	USED
-	0x1bc	500	FREE
5	0x3b0	10	USED
-	0x3ba	70	FREE

---

```
Press Enter to continue... 
```

*Fig 4 : We are going to use this memory state to demonstrate allocation strategies*



[illegible]

*Fig 6 : Malloc 50, on Fig 4 state using best\_fit allocation strategy*

### 3.3 Worst-Fit Allocation

**Definition:**

- Chooses the largest available free block for allocation.

### Characteristics:

- Leaves large remaining free blocks
- May reduce the creation of very small fragments

### Implementation Notes:

- Scans all free blocks
- Selects the block with maximum size

```

Memory Management Simulator
-----
Total      : 1024 bytes
Used       : 204 bytes
Free       : 820 bytes
Allocations : 8 Success / 0 Fail (100.0%)
Ext Frag   : 45.12 %

Memory Map:
[■■■■■■□□□□□□□□□□□□■■■■■■□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□]
Legend: ■ = USED  □ = FREE
-----

Enter command:
> dump
Block Map:
-----
ID      Start      Size      Status
-----
1       0x0         100      USED
-       0x64         300      FREE
3       0x190        44      USED
8       0x1bc        50      USED
-       0x1ee        450      FREE
5       0x3b0        10      USED
-       0x3ba        70      FREE
-----

Press Enter to continue...

```

*Fig 7 : Malloc 50, on Fig 4 state using worst\_fit allocation strategy*

## 4. Freeing Memory and Coalescing

### 4.1 Free Operation

- Memory is freed using free <block\_id>.
- The corresponding block is marked as free.

### 4.2 Coalescing

After freeing a block, the simulator performs **coalescing**:

- Adjacent free blocks are merged into a single larger free block.
- This reduces local external fragmentation without relocating allocated blocks.

#### Key Property:

- Coalescing is a **local operation** and does not move allocated blocks.

[illegible]

*Fig 9 : State after freeing*

- Compaction is an **explicit operation**, triggered by the compact command.
- It is not performed automatically during allocation.

Press Enter to continue...

Press Enter to continue...

[illegible]

- Largest free block
- External fragmentation ratio

External fragmentation is defined as:

$1 - (\text{largest\_free\_block} / \text{total\_free\_memory})$

## 6.2 Stats Update Model

- Statistics are cached in a dedicated Stats structure.
- Stats are updated after every memory-modifying operation:
  - allocation
  - free
  - compaction

### Memory Management Simulator

---

```
Total      : 1024 bytes
Used       : 594 bytes
Free       : 430 bytes
Allocations : 12 Success / 1 Fail (92.3%)
Ext Frag    : 32.56 %
```

*Fig 12 : Dashboard statistics. Fail represents failed attempts to allocate memory*

---

## 7. Visualization and User Interface

### 7.1 Dashboard

The simulator includes a terminal-based dashboard that:

- Continuously displays memory usage
- Visualizes blocks using symbols:
  - ■ for allocated memory
  - □ for free memory

### 7.2 Design Goals

- Always-visible global state
- No hidden memory behavior
- Clear separation between UI and memory logic

## 8. Command Interface

Supported commands include:

- `init memory <size>`
- `set allocator first_fit | best_fit | worst_fit`
- `malloc <size>`
- `free <block_id>`
- `compact`
- `dump` (block-level memory map)
- `help`

The command interface is intentionally minimal and focuses on memory operations rather than user interaction complexity.

---

## 9. Limitations

- No paging or virtual memory support
- No internal fragmentation modeling
- No concurrency or multi-process simulation
- Memory accesses are not timed

These limitations are intentional to keep the focus on contiguous allocation behavior.

---

## 10. Conclusion

This simulator demonstrates the behavior and trade-offs of classic contiguous memory allocation strategies. Through visualization, statistics, and compaction, it provides insight into fragmentation and memory management decisions commonly discussed in operating systems.

Future extensions could include paging, buddy allocation, or integration with cache simulation.