Contiguous Memory

Introduction

This is a simple program that demonstrates how contiguous memory is allocated in C.

Usage

run the following command:

```
make
```

type `help` to see the available commands.

Implementation

Data Structures:

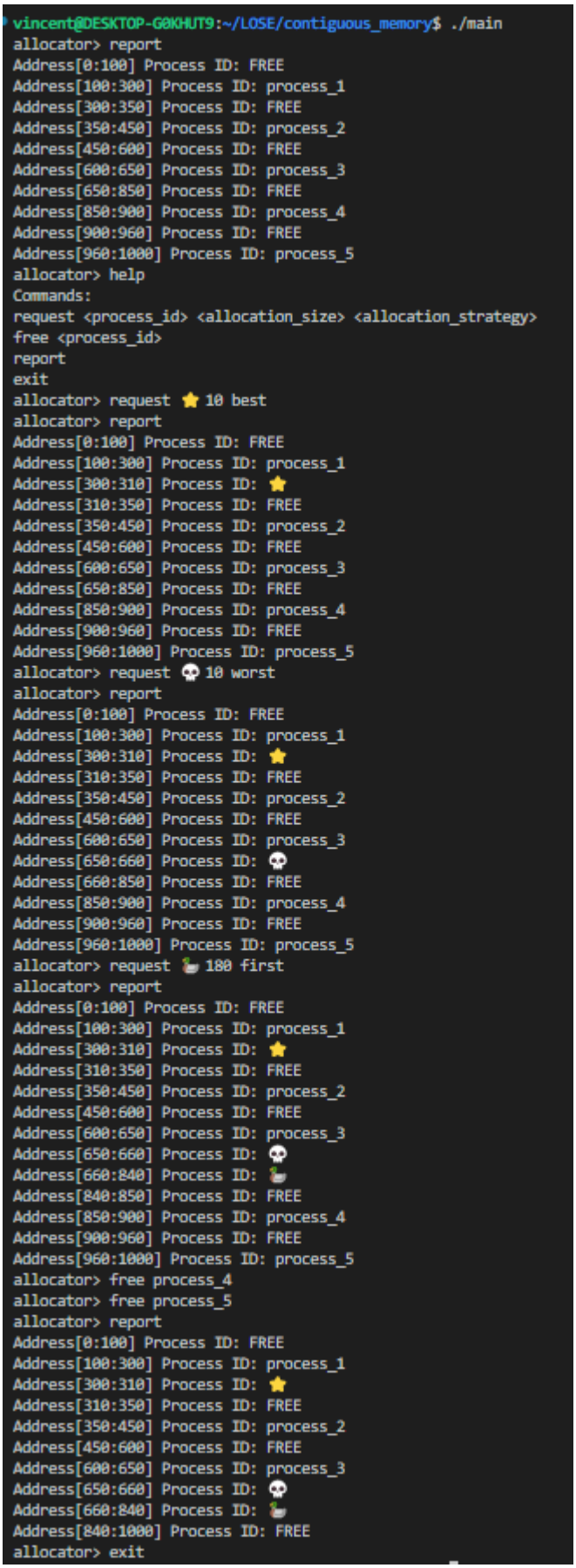
1. **Block Structure (block_t):**
   - Represents a memory block.
   - Contains fields for start address, available space, process ID, and a pointer to the next block.

Functions:

1. **Memory Allocation Strategies:**
   - **First Fit (`find_first_fit`):**
     - Searches for the first available memory block that fits the allocation size.
   - **Best Fit (`find_best_fit`):**
     - Searches for the memory block that best fits the allocation size.
   - **Worst Fit (`find_worst_fit`):**
     - Searches for the memory block that worst fits the allocation size.
2. **free_blocks:**
   - Frees all memory blocks associated with the given process ID.
   - `merge_unused_adjacent_blocks` is called after freeing the blocks.
3. **allocate_blocks:**
   - Allocates memory blocks for the given process ID after the block is found using the allocation strategy.
   - `init_block` is called to initialize the block.
4. **request_block:**
   - Requests memory blocks for the given process ID using specified allocation strategy.

5. **free_list:**
    - free the linked list of blocks.
6. **setup_memory_state:**
    - Initializes the memory state for easy demonstration.
7. **report_memory_usage:**
    - Reports the memory usage.

Screenshots


vincent@DESKTOP-G0KHUT9:~/LOSE/contiguous_memory$ ./main
allocator> report
Address[0:100] Process ID: FREE
Address[100:300] Process ID: process_1
Address[300:350] Process ID: FREE
Address[350:450] Process ID: process_2
Address[450:600] Process ID: FREE
Address[600:650] Process ID: process_3
Address[650:850] Process ID: FREE
Address[850:900] Process ID: process_4
Address[900:960] Process ID: FREE
Address[960:1000] Process ID: process_5
allocator> help
Commands:
request <process_id> <allocation_size> <allocation_strategy>
free <process_id>
report
exit
allocator> request ⭐ 10 best
allocator> report
Address[0:100] Process ID: FREE
Address[100:300] Process ID: process_1
Address[300:310] Process ID: ⭐
Address[310:350] Process ID: FREE
Address[350:450] Process ID: process_2
Address[450:600] Process ID: FREE
Address[600:650] Process ID: process_3
Address[650:850] Process ID: FREE
Address[850:900] Process ID: process_4
Address[900:960] Process ID: FREE
Address[960:1000] Process ID: process_5
allocator> request 💀 10 worst
allocator> report
Address[0:100] Process ID: FREE
Address[100:300] Process ID: process_1
Address[300:310] Process ID: ⭐
Address[310:350] Process ID: FREE
Address[350:450] Process ID: process_2
Address[450:600] Process ID: FREE
Address[600:650] Process ID: process_3
Address[650:660] Process ID: 💀
Address[660:850] Process ID: FREE
Address[850:900] Process ID: process_4
Address[900:960] Process ID: FREE
Address[960:1000] Process ID: process_5
allocator> request 🦆 180 first
allocator> report
Address[0:100] Process ID: FREE
Address[100:300] Process ID: process_1
Address[300:310] Process ID: ⭐
Address[310:350] Process ID: FREE
Address[350:450] Process ID: process_2
Address[450:600] Process ID: FREE
Address[600:650] Process ID: process_3
Address[650:660] Process ID: 💀
Address[660:840] Process ID: 🦆
Address[840:850] Process ID: FREE
Address[850:900] Process ID: process_4
Address[900:960] Process ID: FREE
Address[960:1000] Process ID: process_5
allocator> free process_4
allocator> free process_5
allocator> report
Address[0:100] Process ID: FREE
Address[100:300] Process ID: process_1
Address[300:310] Process ID: ⭐
Address[310:350] Process ID: FREE
Address[350:450] Process ID: process_2
Address[450:600] Process ID: FREE
Address[600:650] Process ID: process_3
Address[650:660] Process ID: 💀
Address[660:840] Process ID: 🦆
Address[840:1000] Process ID: FREE
allocator> exit