Masoud Rafiee-OS (CS409)-Assignment_02

Manual Instruction for Reproducing the Project

1. **Prerequisites:**
   - A C compiler (such as gcc).
   - A Unix-like environment (Linux, macOS, or Wsl for Windows,) is recommended since the code uses system calls like `read` and `write`.
2. **Compilation:**
   Open your terminal and run:

   ```
   gcc -o allocator project.c
   ```

   This compiles the code into an executable named `allocator`.

3. **Running the Program:**
   Execute the program by passing the total memory size (in bytes) as a command-line argument: `./allocator 1048576` (This example uses 1 MB of memory.)
4. **Using the Interactive Commands:**
   At the prompt `allocator>`, you can type commands (each command is followed by pressing Enter):
   - **RQ (Request Memory):**
     Example: `RQ P0 40000 F`
   - This will request 40,000 bytes for process P0 using the First Fit strategy. Use "B" for Best Fit or "W" for Worst Fit.
   - **RL (Release Memory):**
     Example:`RL P0`
   - This command releases the memory associated with process P0.
   - **C (Compact):**
     Type: `C`
   - This command compacts the free memory into one large block.
   - **STAT (Status Report):**
     Type: `STAT`
   - This shows the current state of memory with all allocated and free regions.
   - **X (Exit):**
     Type: `X`
   - This exits the program after cleaning up memory.
5. **Program Behavior:**
   - The program uses a dummy head node and an initial block representing all available memory.
   - Memory blocks are split, merged, and rearranged based on user commands.
   - Each block in the memory map displays its start and end addresses and the process ID (or "Unused" if the block is free).
6. **Cleanup:**
   Before exiting, the program calls `prepare_to_exit` to free all dynamically allocated memory to prevent memory leaks.