Memory Management API - Phase 1 Report

Baran Çimen 26336

My program starts with creating threadIDs array and threads array to store the threads' IDs and theirselves. Then it creates threads and puts them in the threads array, also it puts their IDs in the threadIDs array. Then the program will call init(), which will lock the shared mutex and creates thread semaphores for each thread and put them in the global semlist array. Also the init() will fill the memory with '0's meaning that they are not in use. Last but not least, it will create the server thread and run the server_function on the server thread and unlock the mutex.

The Server thread will run the server_function(), which will run until all other threads are done. While the server is running, it will constantly lock and unlock the mutex, and will take the next node from the front of the queue. It will check if there is enough memory in the memory, and if there is it then the server will change the value of the threads respective thread_message with the first index of the available memory. After it finishes those operations, it will unlock the threads respective semaphore.

The main function will continue with creating threads and after creating them it will join them.

The threads will create a random size of memory, lock the mutex and use my_malloc() to queue the memory request. Then it will be blocked by the its respective semaphore until the server unlocks the semaphore. When the server does, the thread will check if there are enough memory from the thread_message, if there is then it will write it to the memory array with the index it took from the thread_message. If the thread_message is -1, it means there aren't enough memory for this operation and it will print an error message.

When the main function joins all the threads, it will run the dump_memory() function which will print the memory to show the allocations. After that, the main function will print the memory indexes from the thread_messages. Lastly, the main function will close the server by changing the server_on variable and will pop all nodes from the queue, and destroy all semaphores for deallocation.