CS307 HW3 Report

Mehmet Enes Battal – 26354

In this homework a simple memory management API is implemented. The shared memory is an array of integers. An array of integers was used because when the memory was an array of characters, there were conversion/typecasting issues that couldn't be solved. There is a mutex for the control of shared data access and there are semaphores for all the threads which enable the control of those threads. Also, there is an integer array called thread message which indicates if corresponding threads can continue with using the memory or not.

At initialization the memory array is set to all zeros and a semaphore for each thread is initialized. Each thread has its own id from 0 to (number of threads -1) and this id is used to access the thread message and get the right message. Server thread (will be referred as server for simplicity later) is also created at initialization and continues to run in a loop until all other threads with ids finishes. When all threads finish server thread also finishes and the execution is continued by printing the memory array to the console.

Each thread with an id sends a request to server and waits by decrementing a semaphore until server sends a message. This request is added to a queue that is handled by the server and server decides to give permission to use requested memory or not. Each thread's requested size is a random number between 1 and MEMORY_SIZE / 6 and server gives permission if the requested memory size is available by setting the corresponding index of thread message array to the next available index in the memory. If not, the thread message is set to -1 so that when corresponding thread continues its execution it prints a message saying there is not enough memory. Either way the same semaphore is incremented so that thread can continue its execution. If permission is granted, thread continues to write its id to the memory for requested size times.