CS307 HW3 Tuğcan Barbin 25168

Firstly, I used srand(time(NULL)) with <time.h> to create random seed to our rand() function.

Then, i created an array that contains ids of threads starting from 0 and goes to 9.

Then, init()

Then, i created 10 threads by giving them the id from my previous created array , So threads are started to execute the given function which is thread_function. And joined all threads to make sure that main will stop executing and wait for other threads to terminate.

In thread_function, firstly i took the id that comes from create function and converted it int from void. Created random int that between 1 and memorysize/6, and used my_malloc function with parameters of id of that tread and the size that we created randomly.

İn my_malloc,

İ take thead id and size into a temp node, then i locked the mutex and pushed that node into myqueue. So other threads can not acces to shared queue and create a race condition. Then unlocked.

We will turn back to thread_function, and the current thread will down its semaphore to wait untill the server thread gives an Access. İf the thread gains acces, it will start to write to memory that by starting from given index and continues untill it allocates the size of memory that it want to allocate and allocates the id of it.

Server thread executes the server_function which controls the queue. At first i created a variable to control the while loop, a temp node to take a node from queue and an integer index.

It is working with a while loop with a condition that continues untill all threads are done.

İn while i took the front node of queue if queue is not empty, then i popped the element. If there is enough memory i passed the starting index of that avaliable memory through message. If not , then message is -1.

After the control finished server_function ups the given thread's semaphore

Dump memory is printing all the indexes one by one in memory array.