Yiğit Karamanlı

28105

yigitkaramanli@sabanciuniv.edu

LOCKING MECHANISM

In this assignment, I used Pthread locks to assure that concurrency will not cause any data races. I've used coarse grained locking and locked the whole linked list during allocation and freeing operations such that it will assure no data race conditions occur on the linked list object. When a thread calls on one of the myMalloc() or myFree() functions, it must acquire the lock before doing anything, and the lock is released after all changes took place on the linked list and just before returning, thus providing isolation of the threads from each other and guaranteeing the atomicity of the instructions.

IMPLEMENTATIN DETAILS

myMalloc() Function:

The function first starts with acquiring the lock on the linked list and creating a pointer(ptr) that points to the head of the linked list. After that, this pointer iterates over the linked list until finding an available node with enough capacity. If no such node is found, the appropriate error message is displayed, the list is printed, lock is released and the function returns -1. Otherwise the thread checks if the size of the available node is equal to the requested size or not. If so, the 'id' of the node is changed to the requesting thread ID. If the size of the node is larger than the requested size, the node is divided into two partitions: first one having the requested size with ID= tid and second one contains the remaining free space of the original node.

After all operations are done, appropriate message is displayed, the state of the linked list is printed, lock is released and the program returns the starting index of the chunk that is allocated by the thread.

myFree() Function:

The function starts with acquiring the lock on the linked list and initializing two pointers, one for finding the node to be freed and one for checking the node just before it (since the linked list is one ways). After that, the list is iterated over in order to find the allocated node to be freed. If no such node is found, the appropriate error message is displayed, the list is printed, lock is released and the function returns -1. Otherwise, the ID of the node is changed to '-1', and its previous and the next nodes are controlled. If either one of them or both of them is also 'free', they are merged together into one single free node.

After all operations are done, appropriate message is displayed, the state of the linked list is printed, lock is released and the program returns 1.