Atakan SEVİNÇLİ - 29077970382 12.05.2021

Öykü Çise BİLİK -14074215802

CMPE382

Operating Systems

Project #2 : Creating TEDU_alloc

Instructor: Asst. Prof. Tayfun Kucukyilmaz

TA: Bedrettin Çetinkaya

In our project we have worked as a pair programming.Our first step in achieving this goal was to meet as a group in Zoom.We threw out ideas on what we were thinking regarding the project and try to understand expectations from us.During our other meetings, we have discussed our ideas and worked simultaneously.As a group we did not split aspects of a chapter.Instead, we worked on online meeting platforms such as Microsoft Teams and Zoom.We thought we could be faster and more efficient by working like this.We have share our ideas and did not have any difficulties in finding our mistakes by screen sharing and remote control at a certain time every day via Zoom.

Design overview:

Our code includes all of the given methods in the project task. The expectations of the given task are to create own malloc and free methods with a few features such as regulating and monitoring all access to the memory . map between addresses and memory objects. To do these, we should track the regions of memory allocated by TEDU_alloc. So, we need sophisticated data structure. We have used the linked list to perform all methods with extra features.

Thanks to the integer pointer **(int* ptr)** in the node object, we have mapped between addresses and memory objects. Each node can keep the size of the block of addresses it represents, along with the **(int size)** variable. Then the pointer of the node next **(Node *next)** points to the other node object in our linked list When users want to allocate memory, linkedlist finds the first empty memory block. Allocates if there is enough space in that block. Thus, our policy is suitable for first-fit policy.

Complete Specification:

First of all, our *Mem_Inıt* method allocates memory requested from us with the help of **sizeOfRegion** parameter. Then it creates the required data structures (head of the linkedlist) for us to monitor the usage of allocated memory.

TEDU_alloc checks if there is enough space already allocated in memory of the size the user wants.If there is enough space, a node is created.Otherwise it returns NULL. The **int * ptr** variable of the created node maps between addresses and memory objects.This variable also specifies how much space should be kept here along with the given **int size** variable.

The *TEDU_Free* method frees the memory block that the user has previously allocated, just as the standard free method in C.Firstly, the method we created hovers over nodes and checks matching status with ***ptr.** If it matches, then the parameter ***ptr** is in our memory block**.**Finally, our linkedlist removes the corresponding nodes.If the parameter of the method is NULL no operation performed.Thus, we fulfill the requirements of the *TEDU_Free* method.

For our *Mem_GetSize* and *Mem_IsValid* methods, Just like Tedu_free method hovers over nodes and checks matching status with ***ptr.** If there is a match, *Mem_IsValid* method returns 1,otherwise returns 0. If there is a match for *Mem_GetSize* method, return size of the current node for us,otherwise return -1.

Four our last method *TEDU_GetStats* , during the iteration on the linkedlist we can count the number of nodes in our linkedlist,make summation of node's size(used memory) and give the percentage of used memory.Thus we can hold the node with the largest size. Thanks to this data, we can print the memory usage summary of the allocated memory.

Evaluation of the Speed:

We have used linked list as a data structure so;

- Linked list is a dynamic data structure so it can grow and shrink at runtime by allocating and deallocating memory.
- Insertion and deletion of nodes are really easier.
- As size of linked list can increase or decrease at run time so there is no memory wastage.
- In a linked list, traversing the elements or nodes is difficult. We can't access any element at random, like we can with an array.

Although Linked lists are so useful, there may be a loss of performance here because we need to check all the nodes when we need to traverse.The processing of the first fit is fast. The processor assigns the job to the first accessible memory partition.

Known issues and bugs:

There is no known bugs or issues.