12.05.2021
Tanercan ALTUNTAŞ 12595552970
Ahmet Kaan TOPRAKÇIOĞLU 12742035240

Tanercan ALTUNTAŞ 12595552970
Ahmet Kaan TOPRAKÇIOĞLU 12742035240

CMPE 382 PROJECT 2

Operating Systems

Project #2 : Creating TEDU_alloc

INSTRUCTOR: Dr.Tayfun KÜÇÜKYILMAZ

12.05.2021
Tanercan ALTUNTAŞ 12595552970
Ahmet Kaan TOPRAKÇIOĞLU 12742035240

Distribution of Task:

We worked as a couple programming in our project. We had many online interviews with the effect of the pandemic process in order to make our project and our first step to achieve this goal was to organize a goal meeting as a group. We discussed our thoughts on the project and tried to understand what was expected of us. We discussed our ideas in our previous meetings and worked at the same time. We did not separate some parts of an episode as a group. Instead, we used online collaboration tools like Microsoft Teams and Discord. We thought we could work faster and more efficiently by working in this way. With Discord, we were able to share our ideas and find our mistakes easily at a certain time every day by screen sharing and remote control.

Design overview:

In this project, we were implement a memory allocator for programmers. Our memory allocator is more interesting than the traditional malloc( ) and free( ). We first created a “struct” structure for our project. This structure represents the memory (RAM) of the program.

After creating a virtual memory by using "struct" structure in our program, a space is allocated with the dimensions determined by the user. First, virtual memory is a single whole. When the user wants to allocate a space from the memory, the memory is divided into itself scalable. The user allocates the first free part of memory. Thus, our program uses the first fit strategy. While adding elements to our block structure, we use the stack data structure. In the deletion process, any field in the block can be deleted.

Complete Specification

We have mentioned a little bit about our “struct” structure in the Design Overview section. We have 3 variables inside the struct. The first of these is "sizeOf region". This variable holds the capacity of the program's memory. The second is a pointer array named memory. Its type is integer. This variable has an important role in our program. The first part of the variable holds the parts of the program's memory itself. In the second part of the variable, it keeps the addresses of the variables created from the integer pointer type. The last variable is called "part_of_block_size". Its type is integer pointer. This variable holds the size of the pointer variables placed in the memory parts.

We initialize the memory size of the program with the “Mem_Init” function. While determining the memory size, our memory is created as a single part. However, the “part_of_block_size” variable is created as one part. If the user calls the "TEDU_alloc ()" function. First, the size requested by the user from the memory is checked, if there is enough size in the memory, the memory of the program is divided into itself in a scalable way. Also, to avoid unused memories. The size specified by the user is made stable. That is, because our program only operates on variables of integer data type. The size is rounded off to a multiple of 4. Finally, the size that the user wants to allocate is added to the last part created in our memory. In this way, we are using the first fit strategy.

12.05.2021
Tanercan ALTUNTAŞ 12595552970
Ahmet Kaan TOPRAKÇIOĞLU 12742035240

The void * ptr variable included in the TEDU_Free function checks whether the address of each partition in our memory block matches with * ptr and removes the required part in case of matching. The free partition is deleted, and the current memory is updated. While the section to be deleted is searched in the memory block, all sections are checked from the beginning and sequentially in the loop. In case of matching, the loop is exited with the break command. In this way, the loop is get out from unnecessary turns and our program runs more efficiently.

Similar algorithms are used in "Mem_IsValid" and "Mem_GetSize" functions. It is checked whether the address of the variable retrieved from the parameter matches the addresses of the regions inside our memory block. This control algorithm is the same as the search algorithm in the "TEDU_Free" function. In case of matching, the Mem_IsValid function returns 1, the Mem_GetSize function returns the size of the value in the parameter.

Evolution of the Speed of Our Program

A resource allocation scheme (usually for memory). First fit fits data into memory by scanning from the beginning of available memory to the end, until the first free space is founded. First fit is fast in processing. As the processor allocates the nearest available memory partition to the job, it is very fast in execution.

Bugs or Problems:

There are no bugs or problems in our programs.

How to Run

We prepared a “makefile” file, but we encounter an unexpected error. So, you can compile the code with "./compile" and run it.