ITP 30002-02 Operating System, Spring 2020

Homework 5

Jeon, Yeo Hun / 21500630 / 21500630@handong.edu

1. Introduction

In this assignment, I enhanced ‘smalloc’, a simple malloc library, to version 2.0. In the improved library, it supports best-fit strategy to allocate new memory space, and compaction when allocated memory is freed by user. In addition, it also provides API to check dynamic memory usage status of the system and API to reduce program break point. To implement above features, I created some strategy to traverse sm_container_t objects and make connection of the pointer of each object according to given situation.

2. Approach

The first feature of the library is to use best-fit strategy to allocate new memory space. To implement this feature, I modified the original ‘smalloc’ function. the program traverses all the existing sm_container_t objects and find the object who has the smallest size difference with the requested size.

The second feature of the library is to use compaction strategy when user release memory. For this feature, I modified the original ‘sfree’ function. The program checks nearby unused object and collect all the memory size into one object. A thing to be considered is the size of the object. Because of the allocation size is held by the sm_container_t object, the program needs to add up all the object size (32 bytes) when adding up the compacted size.

The third feature is a new API to check current usage of memory that shows amount of memory retained by smalloc, amount of memory currently using and empty. This API is implemented by continuously adding up the retained memory space when smalloc is called to get total retained size. And also grouping and adding all memory space size those have status ‘Busy’ and ‘Unused’ to get current information of the memory.

The fourth feature is a new API to resize the allocated memory space by user-given size. For this, the program can counter several different scenarios. The first case is when there is possible contiguous space to expand. At this moment, the program simply takes the extra memory from the space and increase the size saved in the object. The second case is when there is no free contiguous space after the target memory address. In this case, the program first looks for existing hole that has enough size to allocate new size. If there is a possible hole, the program migrates the original data into the new hole. If there is no hole, which is the third case, the program retains new memory space and allocate new size to migrate the original data.

The last feature is to shrink break point of the heap memory space. This part is implemented by using sbrk() system call with parameter of the total size of current unused memory space. Removing all the unused data from the list and giving negative value of sbrk() system call, it reduces break point location from current location to relative location with given parameter.

3. Evaluation

To evaluate the library, I made some questions.

1. Does the library successfully provide expected functions? This question is answered by run test case with whole functions: smalloc(), sfree(), sshrink(), srealloc(), print_mem_uses(). As the result, I could observe all the functions made expected results as described in previous section.
2. Does the library successfully be utilized with best-fit strategy? To answer this question, I intentionally made situation with several holes, with size of 1500, 1532 and 1032 in sequential manner. And when I allocate new memory with size of 1000, I could observe that hole with 1032 bytes is consumed to make the allocation.
3. Does the library successfully be utilized with compaction? To answer this question, I add multiple sequential sfree() in the test program. As the result, I could observe the continuous unused holes are all merged together.

4. Discussion

The library will be much better if it can find all memory space with ‘Unused’ status and merge them all to make the actual compaction. Even though This will make have overhead to traverse all the object list, there will be an advantage in using smalloc, because it does not need to check all the object but directly access to the compacted hole and make allocation.

5. Conclusion

In this assignment, I implemented improved dynamic allocation library with best-fit strategy and merging adjacent free memory into one object. This library also provides resizing of allocated memory and compress reduce break point as much as possible.