ITP 30002-01 Operating System, Spring 2020

Homework 5

21600212 Nam Jinwoo | 21600212@handong.edu

1. Introduction

In this paper I introduce my own simple heap memory allocation library – smalloc 2.0. This paper describes 5 tasks that I completed and explains approaches to implement 5 tasks.

Task 1: Revise smalloc() from first-fit to best-fit algorithm
Task 2: Revise sfree() to merge adjacent unused containers
Task 3: Add a new API that print out memory uses
Task 4: Add a new API which resizes the memory allocated
Task 5: Add a new API which reduces the break point

Moreover I self-prove that I successfully accomplished the given tasks with test program. At the 4.Discussion, I also discuss possible improvements over smalloc 2.0.

2. Approach

In order to accomplish **Task 1**, I revise smalloc to find minimum space among available space which space is larger than given size. I also made ***test4.c*** program which bigger unused container is on the above. That means first-fit waste memory so it needs to retain more memory later.

```
        if (itr->status == Busy)
                continue ;
        if ((itr->dsize == size) || (size + sizeof(sm_container_t) < itr->dsize)) {
                if(min >= itr->dsize)
```

In order to accomplish **Task 2**, I revise sfree to find all consecutive unused containers. Condition satisfied when unused container has next container which status is unused.

```
for (itr = sm_head.next ; itr != &sm_head ; itr = itr->next) {
    if(itr->status == Unused && itr->next != &sm_head) {
        if(itr->next->status == Unused){
```

In order to accomplish **Task 3**, I print out 1. the amount of memory retained so far, 2. memory allocated at this moment and 3. memory retained but not currently allocated

In order to accomplish **Task 4**, I devide it to 4 cases.

Case 1: newsize is smaller than hole->size
Split the container and sfree() the remainder space.

Case 2: hole->next == &sm_head
Retain more memory, merge them and split it.

Case 3: hole->next->status != Unused || not enough mem
Find best-fit space or retain more memory, merge, split.

Case 4: hole->next->dsize has enough memory

Merge hole and next, and then split hole to new size.

In order to accomplish **Task 5**, I find topmost unused containers and remove. To remove, reconnect previous and next containers first, and then reduce heap size using brk().

```
            int temp = (int)itr->dsize + sizeof(sm_container_t);
            itr->prev->next = itr->next ;
            itr->next->prev = itr->prev ;
            brk(curr_brk - temp) ;
```

3. Evaluation

In order to evaluate tasks, I made test program and run it.

```
================== sm_containers ==================
  0:0x23f5020:  Busy:    2000:00 00 00 00 00 00 00 00
  1:0x23f5810:Unused:    2032:00 00 00 00 00 00 00 00
  2:0x23f6020:  Busy:    2500:00 00 00 00 00 00 00 00
  3:0x23f6a04:Unused:    1532:00 00 00 00 00 00 00 00

smalloc(1000):0x23f6a04
================== sm_containers ==================
  0:0x23f5020:  Busy:    2000:00 00 00 00 00 00 00 00
  1:0x23f5810:Unused:    2032:00 00 00 00 00 00 00 00
  2:0x23f6020:  Busy:    2500:00 00 00 00 00 00 00 00
  3:0x23f6a04:  Busy:    1000:00 00 00 00 00 00 00 00
  4:0x23f6e0c:Unused:     500:00 00 00 00 00 00 00 00
```

Task 1 - Successfully find best-fit unused container.

```
sfree(0x23f6020)
================== sm_containers ==================
  0:0x23f5020:  Busy:    2000:00 00 00 00 00 00 00 00
  1:0x23f5810:Unused:    4564:00 00 00 00 00 00 00 00
  2:0x23f6a04:  Busy:    1000:00 00 00 00 00 00 00 00
  3:0x23f6e0c:Unused:     500:00 00 00 00 00 00 00 00
```

Task 2 – Successfully free and merge unused containers.

```
================== mem_uses ==================
The amount of memory retained so far: 8192
The amount of memory allocated at this moment: 3000
The amount of memory retained but not allocated: 5064
```

Task 3 – Successfully print out memory uses informations.

```
================== sm_containers ==================
  0:0x239e020:  Busy:    2000:00 00 00 00 00 00 00 00
  1:0x239e810:  Busy:    1000:00 00 00 00 00 00 00 00
  2:0x239ec18:Unused:    3532:00 00 00 00 00 00 00 00
  3:0x239fa04:  Busy:    1000:00 00 00 00 00 00 00 00
  4:0x239fe0c:Unused:     500:00 00 00 00 00 00 00 00

srealloc(0x239e020, 3000)
================== sm_containers ==================
  0:0x239e020:Unused:    2000:00 00 00 00 00 00 00 00
  1:0x239e810:  Busy:    1000:00 00 00 00 00 00 00 00
  2:0x239ec18:  Busy:    3000:00 00 00 00 00 00 00 00
  3:0x239f7f0:Unused:     500:00 00 00 00 00 00 00 00
  4:0x239fa04:  Busy:    1000:00 00 00 00 00 00 00 00
  5:0x239fe0c:Unused:     500:00 00 00 00 00 00 00 00
```

Task 4 – Successfully realloc new size. This example case is task4-case3 (next container status is Busy.)

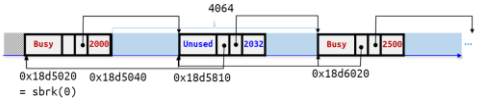
```
===== Sshinked Heap Memory =====
================== sm_containers ==================
  0:0x176e810:  Busy:    1000:00 00 00 00 00 00 00 00
  1:0x176ec18:  Busy:    3000:00 00 00 00 00 00 00 00
  2:0x176f7f0:Unused:     500:00 00 00 00 00 00 00 00
  3:0x176fa04:  Busy:    1000:00 00 00 00 00 00 00 00
  4:0x176fe0c:Unused:     500:00 00 00 00 00 00 00 00
```

Task 5 – Successfully sshrink topmost unused containers.

4. Discussion

I tried to migrate data when changing pointer, but I failed. I would like to find out how to transfer data in the future.

In order to improve over smalloc 2.0, I want to visualize containers' relationship with their memory size like below.



And also worst-fit alogorithm (which find largest unused container) can be added for academic purpose.

5. Conclusion

In this paper I introduce smalloc 2.0 API. Through this assingment I could find that I didn't fully understand memory structure. However, now I can say I understand Heap memory and how it works. Especially, handling memory while protecting data was very useful exeperience.