ITP 30002-02 Operating System, Spring 2020

Homework 5

Eunjun Jang 21800633, jangej1031@naver.com

1. Introduction

In this paper, I will introduce a simple dynamic memory allocation library and its new version 2.0. In version 1.0, basic memory allocation and deallocation functions are implemented. Furthermore, some utility function print_sm_containers is implemented. At the new version of smalloc, new features are available. First, the first feature is a change from first fit to best fit. The old smalloc was implemented with the first fit algorithm. In contrast, the new smalloc function now uses the best fit algorithm. Also, the new sfree function does the extra work of merging unused nearby memory. New functions have been added. The print_mem_uses function tells you about the memory usage. The srealloc function is a function that extends existing memory. The srealloc function extends the existing memory and reallocates it. The sshrink function is a function that compresses memory by lowering the upper limit of the heap if there is unused memory at the end of the heap. I will introduce some tests to show working properly and discuss with further works at the version 3.0.

2. Approach

First of all, to implement best fit in smalloc, I have slightly modified the function to find the smallest allocable memory. Additional merging unused neighbor memory have added at sfree routine. Additional merging unused neighbor memory have added at sfree routine. If there is unused memory on the left, the function tries to merge it to the left. Conversely, if it is right, the function tries to merge it to the right. The print_mem_uses prints three memory allocation data: sum of all allocated memory, sum of busy memory size and sum of unused memory size. Iteration and adding each type by type are very simple solution to acquire these memory data. The srealloc is implemented in this way. First it finds current memory node. After that, it checks next node. If next node is unused and enough size to allocate, it splits next node and merges next node to current node. Otherwise, it calls smalloc to find another available memory and calls memcpy to copy existing data. Finally, sshrink function is implemented to adjust top of the heap point to start of the unused node. While iterating in reverse direction, iteration stops at first busy node appearing. After that, brk function is called to resize heap size with the node address.

3. Evaluation

In this section, I will show the test #4 to compare first fit and best fit.

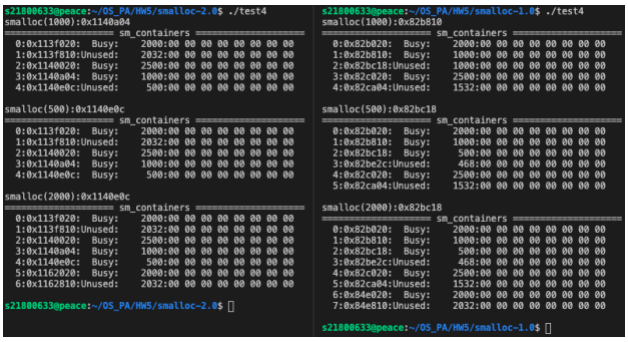

Figure 1. Test4.c (Left: Best fit, Right: First fit)

As you can see in Figure 1, Best fit memory fragmentation is less than first fit one. Memory allocation bytes sequence is following: <2000, 2500, 1000, 500, 2000>

When allocating 1000 bytes, best fit allocated 1000 bytes in index 3 node. But first fit allocated 1000 bytes in index 1 node. Allocating 500 bytes and last 2000 bytes also show difference.

4. Discussion

In this section, I will discuss further version of this smalloc library 3.0. First thing is to implement worst fit algorithm. Implementing the worst-fit algorithm can give users one way to manage heap memory more efficiently. Furthermore, making an API to switch the algorithms in the runtime can provide more powerful options to user to manage heap memory. And it will be good to implement heap memory dump feature. More future version of smalloc can provide simple implementation of reference

5. Conclusion

I discussed smalloc library 2.0 version implementations and also showed the future works. Best fit algorithm and printing mem uses is simply done with iterating all nodes in sequence. And sfree is done with merging adjacency memories. I showed you the test #4 to compare algorithms and talked about future works.