Participants:
Luka Spaic
James Flemings
Emails:
lspaic@alaska.edu
jbflemings2@alaska.edu

Assignment 2 Report

Design Decisions:

- We added a few helper functions to the program such as new_block, get_block, add_block, split_block, and remove_block. These functions allowed us to create a memory map, add a new block of memory, get a block of memory, split a block, and remove a block of memory. Once we were able to implement all of these helper functions, our implementations of malloc, calloc, realloc, and free were able to work correctly.

Choices made:

- The choices we made were based around what we knew we had to implement. So for example, we knew we had to start by creating a map of memory. Thus we wrote the new_block function that added a new block of memory by using mmap and added that block to the linked list. Once we had that we needed to be able to add new blocks which is what add_block is. This function adds the new block to the linked list and is sorted in ascending order by memory address. At this point we also had to be able to get the blocks, thus we wrote get_block. Get_block finds a pointer in a block of memory that has enough length to cover the requested size. In this function, we also had to implement the function that's called split_block. This is used when there is enough memory available in the block that the block can be split. This function splits the block of memory into two pieces. The left side is the pointer with the size that was requested and the right side is the new pointer with its length equal to the remaining available memory. We also had the __free_impl function which is where we're supposed to remove the block. Thus we wrote remove_block for that. This function receives a pointer to a block and either merges it with the left or right pointer, or becomes unmapped. The function merges with the next pointer if it is free and in the same block of memory as the pointer. Next, when all the memory in the current block is free, it then unmaps.

Issues:

- As far as the issues we had while tackling this assignment, we were able to run ls but we had some issues with gcc. Gcc seemed to hang when it was ran. We suspected that it was caused from an endless loop from the program mistakenly creating a circular linked list. Further investigation on this suspicion led us to discover a segmentation fault with some of our pointer arithmetic. Once fixing this pointer arithmetic, running it with gcc worked correctly. Also, Libreoffice doesn't run properly with our memory implementation, but we think that has to do with the threads being used in memory.c. We spoke with Dr. Lauter about it and he verified our claim.

Testing that we ran:

- For testing we performed the example that was shown in class which was the test.c file, ls, firefox, top, and gcc.