Pedram Razavi
Rishabh Kabra

6.172 Project 3.1 Write-up

Overview

We implemented the standard C dynamic memory-allocation functions with the goals of maximizing our heap utilization and throughput.

Data structures

Each allocated memory unit is stamped with a MemoryBlock header to its left and a MemoryBlockFooter to the right. Firstly, the header keeps track of whether a block is a freed unit or an allocated one. Next, the header and footer both keep a record of the size of the whole block so as to allow easy navigation in both directions over the memory space. Lastly, the header stores a nextFreeBlock pointer and a previousFreeBlock pointer, which are used to form the binned doubly-linked free list that the memory block may belong to.

Each bin stores free memory blocks of some particular range of sizes (using the aforementioned doubly-linked free list representation). The function that decides what sizes go into which bin was tuned to balance the load of the trace files we were given.

Algorithm for malloc

We look through the bin corresponding to the size we're asked to allocate to see if we have a freed memory block that can be recycled. If there isn't one, we look through bins containing free blocks of greater sizes. Any available free block is split into a block of the size we need to allocate and another free block that we simply re-bin. If we don't have any free blocks we can use, we simply call mem_sbrk to ask for space.

Algorithm for free

Before we mark a block as free and re-bin it, we check to see if it can be coalesced with any adjacent free blocks. We remove any coalesced blocks from the binned lists they belong to and finally assign the consolidated block to a single bin.

Algorithm for realloc

Instead of simply resorting to malloc and free to reallocate a block, we check to see if the new requested block size is actually smaller than the existing block size, in which case we simply truncate the existing memory block. If not, we check to see if the block that needs to be reallocated is next to a free block which it can coalesce for the reallocation, which also saves us the need to copy memory contents to a new location using the expensive std::memcpy method. If none of these cases are applicable, we have to use malloc and free as before.