++Malloc

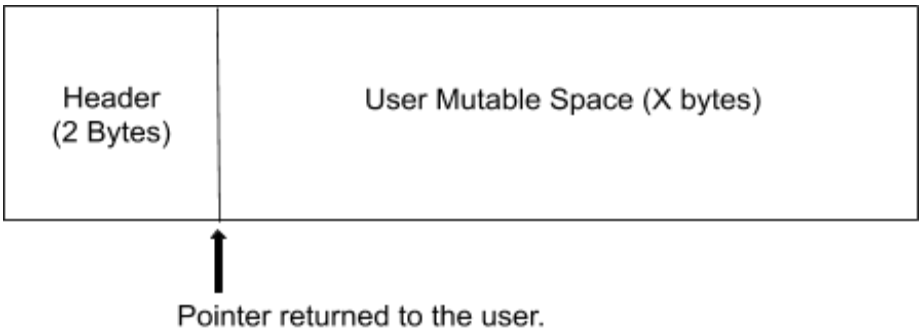
Systems Programming | Section 5

Names:
Christopher Naporlee - cmn134
Michael Nelli - mrn73

Description & Design

--MyMalloc--

Our implementation of **malloc** uses 2 bytes to denote the size of blocks and if they are free or not. 15 bits are denoted just for size, and the final bit is used to denote if the block is free or not. A basic model of how each block of size X looks in memory is:

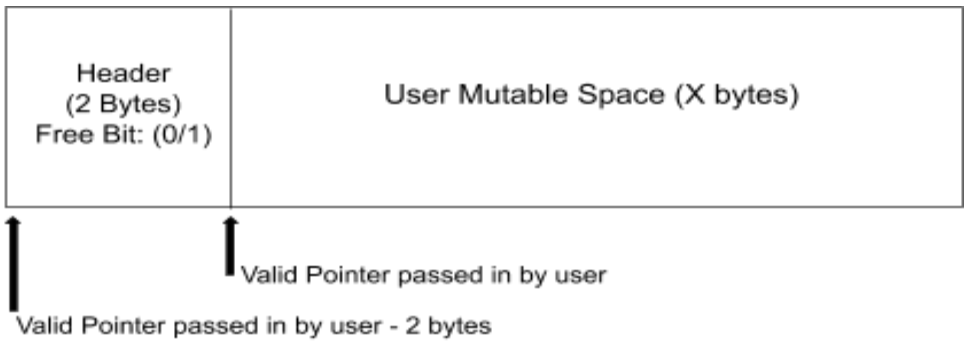


This is efficient because we only keep track of two necessary fields, which blocks are open to be used and the size of each block. This allows the largest block to be requested by the user (given the heap being 4096 bytes total) to be 4094 continuous bytes. The use of the header allows us to use pointer arithmetic to traverse our block of memory going from block to block using the sizes given to us by the header. Each block added up creates a heap that looks like:

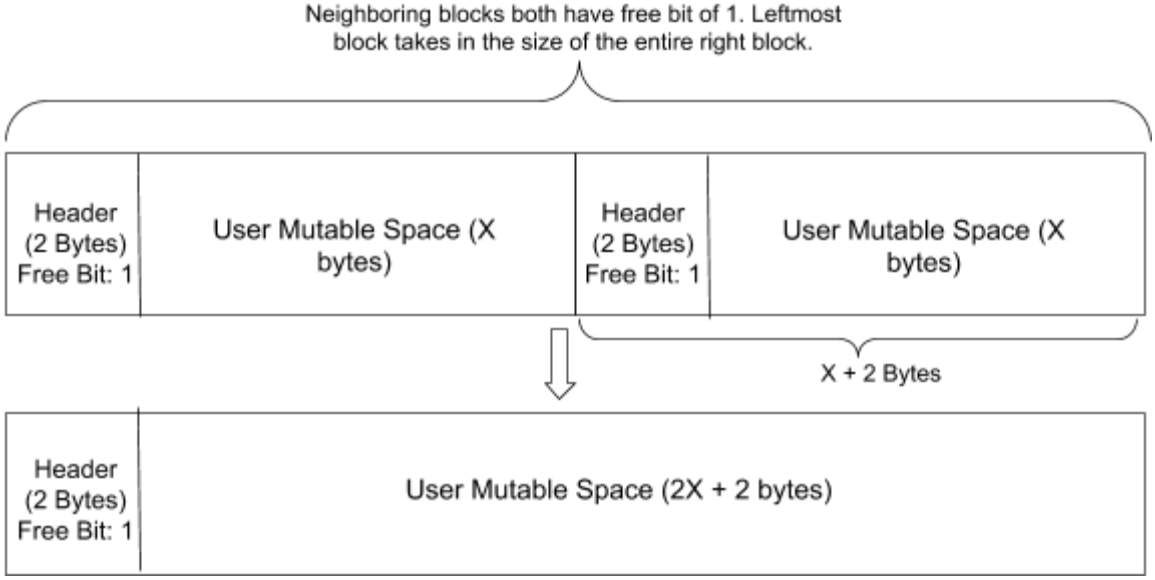
When the user invokes mymalloc, it will go through all the blocks, looking for free ones that are equal to or larger than the requested size. If there are no blocks that match this the user will be warned of the issue (along with the file and line number where mymalloc was invoked) and a NULL pointer shall be returned.

--MyFree--

Our implementation of **free** builds off of the systems implemented from mymalloc. When myfree is invoked it goes through 3 main checks first: is the pointer NULL, is the pointer in range of our heap, and is the pointer pointing to a position that mymalloc returned to the user. If any of these checks fail the user will be warned of the issue (along with the file and line number where myfree was invoked) and will prematurely return. If all three of these cases passed, then we can simply go back two bytes to access our header data. From here we can access our 'free' bit (previously set by mymalloc). If the bit is already set as 1, then we know the pointer is being redundantly free'd, so warn the user and return. Otherwise, set the 'free' bit to 1, to allow this block to be repurposed by mymalloc in the future.



After each successful block that is free'd, myfree will then call *coalesce_blocks*. Starting from the beginning of the heap, *coalesce_blocks* will search for and combine all neighboring free blocks into one full block.



This allows smaller free'd blocks to be combined to form big blocks for mymalloc to use for larger malloc calls.