Jasmin Kaur(jk1539), Jonathan Delin (jd1274)

Systems Programming

Professor Menendez

25 October 2020

Asst1

This assignment required us to implement an advanced version of malloc() and free() that could detect certain errors that those functions could not. To simulate memory, we used an array (myblock[4096]). The mymalloc() function uses a “first free” algorithm to find blocks of the simulated memory to allocate data. This function will return pointers to the array whereas the myfree() function changes pointers to NULL to allow the memory to be reclaimed and used for another purpose.

Errors that we had to detect that malloc and free could not are: free()ing addresses that are not pointers, free()ing pointers that weren’t allocated, free()ing the same pointer redundantly, and allocating more data than the memory can handle. Instead of printing segmentation fault as malloc() and free() would, our functions print what is causing the error (using if-else statements to check for these errors) , and prints the line where the error is occurring. In addition, where free() exits a program when asked to free memory that hasn’t been allocated, myfree() will print why the memory cannot be freed.

To create mymalloc() and myfree(), we implemented a linked list that holds the address, size and index of the data, and also a pointer to the next node that contains the data for the next block in memory. mymalloc() begins with the first block of the array. Using the first-fit algorithm, if it can allocate the requested number of bytes, it will use the first block large enough that it comes across. If there is space left over from the block, a new block is created. For

myfree(), first the function checks for the standard errors of if the pointer is null, redundant, or not even a pointer. Then, it will skip over the current block and set the previous block->next equal to the pointer that current block->next points to. After removing the pointer from the list, it sets the pointer to NULL.

A few aspects we struggled with were segmentation faults when allocating memory for our pointer structs, tracking the blocks of memory, and how to free the pointers in the myfree()function.

Our output is as follows: