Treasa Jose / Sagnik Mukherjee **Oct 20, 2020**

CS 214: Systems Programming, Fall 2020 Assignment 1: ++Malloc

In this assignment we look further into the usage of malloc() and free() and how they aid in dynamic memory allocation. We can better understand the syntax of malloc and know how it returns a pointer value. The free() function indicates that the memory allocation can be freed from the current pointer once its use is complete.

mymalloc.c

This source file contains the primary functions of mymalloc() and myfree(). Along with these we used other functions like sizeSet(), sizeRead() and getFreeIndex() that help in placing the pointer at correct index addresses and adjusting the pointer location once mymalloc() or myfree() is called. The mymalloc() function takes in size, file name and line number as parameters. Checks for any errors like insufficient memory and size being 0. If no errors are encountered the function allocates the specified size (bytes) from the array memblock[4096]. The myfree() accepts a pointer, file name and line number as parameters and frees the malloc()ed memory. The function catches errors like null pointer, un-malloc()ed pointer and redundant myfree() calls. If no error is encountered the specified pointer sets the address for the malloc()ed memory to be freed.

mymalloc.h

Header file contains the definitions

#define malloc (x) mymalloc (x, __FILE__, __LINE__)

#define free (x) myfree (x, __FILE__, __LINE__)

memgrind.c

This file contains the workloads A – E which tests the mymalloc() and myfree() functions. Workloads D and E are explained elaborately in testcases.txt. The output of this file prints the errors encountered during execution, the time elapsed for each workload and their average time.

Makefile

Having looked through the GNU documentation, we created the makefile using @echo commands and used check flags to compile mymalloc.c making the corresponding object file mymalloc.o to create the final executable memgrind. The .o is the library file that memgrind.c is referencing so it can build off that and produce an executable that incorporates both the codes of memgrind.c and mymalloc.c