

CS214 Assignment 1

Patrick Nogaj, Alborz Jelvani

October 25th, 2020

Abstract

This document is project assignment 1: ++Malloc for Systems Programming (01:198:214) at Rutgers University - Fall 2020.

1 Setup your Environment

Within Asst0.tgz file, we have four files that are utilized to run ++Malloc.

mymalloc.c: This is the file that has our methods of *mymalloc()* and *myfree()* to demonstrate the purpose of this assignment.

mymalloc.h: This is a header file which allows us to link methods between files.

memgrind.c: This is the primary testing environment for our program. Within this file, we have five different tests and our output to demonstrate the overall efficiency of our program.

Makefile: This file allows us to compile and remove the program.

To begin: place mymalloc.c, mymalloc.h, memgrind.c, and Makefile all into the same directory on the iLab machine. Within terminal, type "*make*" which will generate an executable called 'memgrind'.

To clean the document, you can type "*make clean*" in the terminal, and this will remove the executable memgrind, and any associating files that are related to running memgrind.

2 Running ++Malloc

To run this program, we can type ".*/memgrind*" into terminal after we have used the make command in the previous section. Arguments are not required to run this program, however, you may utilize "> *output.txt*" to store the output into a file called *output.txt* which would be in the same directory as the program rather than looking at the output in terminal.

TEST	MEAN	SLOWEST	TOTAL
A	5µs	7µs	263µs
B	99µs	114µs	4997µs
C	35µs	43µs	1791µs
D	0µs	1µs	5µs
E	59µs	66µs	2963µs

Once ran, we can expect some output generated that will demonstrate the overall effectiveness of the program. Below is a description of each test labeled from A to E to denote what we were testing with the program.

Test A: For 120 iterations, we had to malloc() 1 byte, and immediately free it.

Test B: For 120 iterations, fill up an array where each index of the array is a single byte that has been malloc(). Once the array has been filled, iterate through the array, and free each 1 byte pointer.

Test C: For 240 iterations, randomly choose between malloc() or free() a single byte; however, there are restrictions. Our first restriction: you cannot free() a byte that has not been malloc'd. Our second restriction: once 120 bytes have been malloc'd, immediately free the remainder of the bytes.

Test D: This workload is meant to test failure cases that would traditionally cause conflicts with malloc/free. We utilized test cases provided from Asst1.pdf to denote the following issues that may arise with user input.

Test E: In workload E, we are looking to test the efficiency of our merging, as it is important for malloc to handle merging of blocks that are not utilized. Please view *testcases.txt* for a more elaborate explanation of workload E.