Building a Better Malloc & Free

Arbazkhan Pathan - ap1415
Griffin - gal69

Systems Programming Spring 2019

Outline:
This projects goal was to create a better malloc and free function then the C library has. This project was done using the following important file:

- Mymalloc.c : this is where we implemented our mymalloc. We created our own malloc inside this file using an array composed of 4096 (integer).
- Mymalloc.h : Inside of this file we have all of the macro definitions, prototypes, as well as any struct definitions that we would need.
- Memgrind.c : This file contains test cases for us to test our code. It acts as a sort of driver to run our “free” and “malloc”.

Design:
For our design we created mymalloc using an array and we stored our metadata in their as well. mymalloc () looked for opportunities for when the user asked for space to store their data.

We also had a few preliminary tests before starting our mymalloc such as testing if the user asked for space greater then we have to give out.

Our code also has a checkIfFree function that checks if the space we are allocating is being used. If it is then we cannot give that space away because the space is being used. Oppositely we have a writeOver function that will overwrite the space if the user wants to.

Our cleanUp() function resets the data so that it can be clean for the next opportunity to run.

Of course since we have a malloc function, we will need a free function. The free function is very similar to how the actual free works in C. If a piece of data needs to be freed then we will clear that data get it ready to be used again in another malloc call assuming that there is enough space.

Challenges Faces
We faced a few challenges while writing our program. One of the biggest problems was that it was difficult to get the time of our code. We first tried to use the time.h in our code, however for some reason it resulted in all zeros for our time, and that makes no sense. Then we decided to use the gettomeofday function instead to get the time of our program. After finnicking with time.h for a while we decided it was either to bail on that idea entirely.

Findings:

Here are our findings:

Time for A: 1847.000000 mean time (microseconds): 1.847000
Time for B: 1317.000000 mean time (microseconds): 1.317000
Time for C: 0.000000 mean time (microseconds): 0.000000
Time for D: 1096.000000 mean time (microseconds): 1.096000
Time for E: 0.000000 mean time (microseconds): 0.000000
Time for F: 1545965950725615 mean time (microseconds): 15459659.507256
total time taken in main (microseconds) : **0.410000**