← Project 2: Memory Allocator

Due by midnight, Saturday 10/13 (or late on Sunday)

(project concept thanks to Dr. Misurda.)

In this project, you will be implementing a memory allocator. You'll be making your own versions of the **malloc** and **free** functions! Yay! Fun!

Materials

You are given the following source files to begin with. Right click and download these files. **Read the comments inside them thoroughly before getting started.**

- mymalloc.h
- mymalloc.c
- mydriver.c
- Makefile
- bigdriver.c

`mymalloc.c` **is where you'll be writing all your code for this project.**

The `mydriver.c` file contains a `main` function for you to expand upon. By default it just allocates and frees a single block of memory. As you work on your allocator, you can add more testing code to this driver program to make sure it works.

`Makefile` is a makefile to make your driver, and eventually, other test file(s) that I give you.

The `bigdriver.c` file is a driver I made to test your allocator more thoroughly. **READ THE COMMENTS INSIDE IT THOROUGHLY TO HELP DEBUG ANY ISSUES THAT COME UP.**

> Although `bigdriver.c` does a lot of tests, it might not catch all bugs. Look at it closely and add tests if you think it's not thorough enough. You can change it and submit it along with your `mydriver.c`.

Building and running

To build everything, use the following command line:

```
make mydriver
```

> **Ignoring warnings in this project would be a very, very bad idea.** The `Makefile` uses `-Wall -Werror`. Do not remove these. *Fix all the warnings. SERIOUSLY.*

Then, you should be able to:

```
./mydriver
```

You can do the same with `bigdriver`:

```
make bigdriver
./bigdriver
```

Your Task

You will write `my_malloc` and `my_free` which implement a simple memory allocator using a linked list and a **next-fit** allocation scheme. The performance won't be great, but hey, it's a class project, not a AAA game engine.

Your `my_malloc` function will:

- Round up the size of the requested allocation to a certain size (I gave you this)
- Try to find a block to reuse, using **next-fit** allocation
- If it found a block, and that block can be split into two smaller blocks, do so
- If it couldn't find a block, expand the heap by moving the break with `sbrk()`
- Mark it as used
- Return the pointer to the data part (after the header)

Your `my_free` function will:

- Figure out the pointer to the header
- Mark the block free

- Coalesce it with any neighboring blocks on either side
- If it's the last block on the heap, contract the heap by moving the break with `sbrk()`

Click here for details on what you have to do.

Requirements and Grading

Basics [30]

- **[6]** Submitted properly
- **[4]** Compiles with the `Makefile` given
- **[5]** Coding style *(don't make two enormous functions, y'all.)*
- **[5]** Properly maintains a linked list of blocks
- **[10]** Successfully completes provided test cases
  - We *will* test your code with another driver besides the one we've given you.

Allocation [40]

- **[10]** Uses **next-fit** to find the next available block
- **[10]** Splits blocks when big enough to be split
- **[10]** If no free block is big enough, expands the heap with **sbrk**
- **[10]** Always returns a valid pointer to a new or previously-unused block!

Deallocation [30]

- **[10]** Freed blocks **can be reused on subsequent allocations**
- **[10]** Coalesces neighboring free blocks
- **[10]** Contracts the heap with **brk** or **sbrk** when freeing the last block on the heap

Extra Credit [20] - that's 2% of your final grade

While next-fit *works,* it's still worst-case linear time to find a block. You can do better by **using a more advanced data structure to store the free blocks.** It's up to you to choose what data structure to use - just as long as it results in **better than linear time** for allocation.

Submission

Follow this checklist:

1. Run `make clean` to remove any executables.
2. Include the following files in your `proj3` directory:
   - `mymalloc.h`
   - `mymalloc.c`
   - `mydriver.c`
   - `Makefile`
   - `bigdriver.c`
   - a `README.txt` *text file* which contains any notes that might be helpful to the grader
     - such as: is anything not working? did you do any extra credit? etc.
3. Did you make a copy of that directory just in case?
4. Did you put your full name and username at the top of each file in comments?

> Name your file `username_proj3.tar.gz` like `jfb42_proj3.tar.gz`.

Now you can **submit using the same directions as lab1**.