Programming Assignment 4: Heap Management

Due: Wednesday April 24 2019 5:30PM

Description

In this assignment you will build your own implementation of malloc and free. That is, you will need to implement a library that interacts with the operating system to perform heap management on behalf of a user process as demonstrated in class.

The code you submit for this assignment will be verified against a database consisting of kernel source, github code, stackoverflow, previous student's submissions and other internet resources. Code that is not 100% your own code will result in a grade of 0 and referral to the Office of Student Conduct.

You may complete this assignment in groups of two or by yourself. If you wish to be in a group of two the group leader must email me your group member's names **by April 15, 2019**. Your email must have the subject line "3320 [Section #] Project 4 Group" where section number is 001 or 003. (003 is the 5:30pm class, 001 is 7:00pm)

This project must be completed, in C, on omega.uta.edu. Windows does not support the sbrk() system call and MacOS/OSX's implementation of shared libraries is unconventional.

Getting the Source

The source code for this assignment may be found at: https://github.com/CSE3320/Heap-Assignment

Building and Running the Code

The code compiles into four shared libraries and four test programs. To build the code, change to your top level assignment directory and type:

```
make
```

Once you have the library, you can use it to override the existing malloc by using LD_PRELOAD:

```
$ env LD_PRELOAD=lib/libmalloc-ff.so cat README.md
```

or

```
$ env LD_PRELOAD=lib/libmalloc-ff.so tests/test1
```
To run the other heap management schemes replace `libmalloc-ff.so` with the appropriate library:

```
Best-Fit:  libmalloc-bf.so
First-Fit: libmalloc-ff.so
Next-Fit:  libmalloc-nf.so
Worst-Fit: libmalloc-wf.so
```

Program Requirements

Using the framework of malloc and free provided on the course github repository:

1. Implement splitting and coalescing of free blocks. If two free blocks are adjacent then combine them. If a free block is larger than the requested size then split the block into two.

2. Implement three additional heap management strategies: Next Fit, Worst Fit, Best Fit (First Fit has already been implemented for you).

3. Counters exist in the code for tracking of the following events:

- Number of times the user calls malloc successfully
- Number of times the user calls free successfully
- Number of times we reuse an existing block
- Number of times we request a new block
- Number of times we split a block
- Number of times we coalesce blocks
- Number blocks in free list
- Total amount of memory requested
- Maximum size of the heap

The code will print these statistics upon exit and should look like this:

```
mallocs:   8
frees:     8
reuses:    1
grows:     5
splits:    1
coalesces: 1
blocks:    5
requested: 7298
max heap:  4096
```

You will need to increment these counters where appropriate.

4. Four test programs are provided to help debug your code. They are located in the tests directory.

5. Implement realloc and calloc:

```
void *calloc(size_t nmemb, size_t size);
void *realloc(void *ptr, size_t size);
```

How to submit homework

1. Submit a gzipped tarball of your source code to blackboard.

Grading

The assignment will be graded out of 100 points. Compiler warnings are there to tell you something is not correct. Pay attention to them and you will save yourself a lot of late nights debugging code. Code that does not compile will earn 0.

Your code will be compiled with the provided makefile on omega.uta.edu .