Programming Assignment 4: Heap Management

Due: December 3 2018 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.

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 November 26th, 2018**. Your email must have the subject line "3320 [Section #] Project 4 Group" where section number is 002 or 003. (003 is the 5:30pm class, 002 is 7:00pm)

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.

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

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 (75pts)

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. Create benchmarks to analyze the metrics above for the different strategies for the five benchmark programs to be posted in the repo.

Report Requirements (25pts)

Each team will be required to provide a written report in PDF format that discusses:.

- Benchmarks: A discussion on how you benchmarked your library and what the results were. You should address:
  - Which heap management strategy does the best job of reusing free blocks? Which one is the worst?
  - Which heap management strategy requires the least amount of heap space? Which one is the worst?
  - Which heap management strategy allows for the most splits? most coalescing? least?
  - Which heap management strategy was the fastest? slowest?
  - Consider the benchmark programs.
    - Which one requires the most mallocs? frees?
    - Which one requests the most amount of space?
    - Which one requires the largest heap?

- You must provide graphs of data and diagrams to backup your conclusions.

- Analysis: A discussion of the following questions:
  - Which heap management strategy suffers the most from fragmentation (what type)?
  - Which heap management strategy is the best?
  - You should provide evidence for your choices.
- Summary: A summary of what you learned about memory management.
- Note, you must incorporate images, graphs, diagrams and other visual elements as part of your report.
- Your report must be in PDF format.

Extra Credit (10pts)

- 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 and the PDF on blackboard.