CS232 Operating Systems
Assignment 03: Implementing memory management routines
Due : 23h55 Monday 11th November, 2019.

CS Program
Habib University

Fall 2019

1 Objectives

(a) Test the student's dynamic memory management concepts.

(b) Test the student's ability to write clean readable code.

(c) Test the student's ability to use Linux System Calls.

(d) Test the student's ability do error handling in code.

(e) Test the student's ability to follow written instructions.

2 Description

In this assignment you will be using concepts from the book chapter titled **Free Space Management** to implement routines which will provide dynamic memory management capabilities to the user.

3 Tasks

Implement the following functions with the described behaviour.

- **int my_init():** this function should initialize a region of memory in 1MB in your program. You will use this block of memory as your heap and manage it dynamically. It should return if it is successfully able to allocate the region of memory and 0 on error. Use the *mmap()* system call for allocating memory.
- **void * my_malloc(int size):** this function should mimic the behaviour of malloc(). It should allocate a block of *size* bytes from the *your heap*, i.e. the block of memory that had been called allocated by your call to *my_init()*, and return a pointer to the start of the newly allocated memory. It should return NULL on error.
- **void my_free (void *):** this function should take a pointer previouly allocated by my_malloc() and free it i.e., return the memory pointed to by this pointer to the pool of memory you will be maintaining internally so that it can be used in a subsequent call to my_malloc(). In case it is passed an invalid pointer it should exit the program by displaying a suitable error message.

- **void * my_calloc(int num, int size):** this should mimic the behaviour of libc calloc() function.
- **void * my_realloc(void *, int size):** this should mimic the behaviour of libc realloc() function.
- **void my_coalesce():** this should coalesce the fragmented free list as described in the book chapter. You should call this function internally from time to time to de-fragment your heap. Choose a judicious time when you think would be appropriate to call this function.
- **void my_showfreelist():** this function should list all the nodes in the free list with the following information for each node in a single line separated by semicolons:
  node_no.:node_size:node_start_address
- **void my_uninit():** this will be the very last function called by the user. This should deallocate the memory block that was allocated by your my_init() function and do any other housekeeping required by your program.

You not use any memory allocation deallocation functions inside your program. The only such function you are allowed is the mmap() call and that too only inside your my_init() function.

You will maintain a dynamic list of the free memory sections in your *heap* i.e. the block of memory allocated inside my_init(). You cannot use any extra memory (dynamic or static) for this list. It has to be maintained *inside* your heap as we studied in the book. The only storage you should use outside the *heap* is the *head* pointer you'll have to point to the start of the list.

Pay special attention to corner cases, error cases, cases which may result in memory leaks on your behalf.

The typical flow of the user who would use your functions would be:

- call *my_init()*.
- call any of the other functions any number of the time.
- call *my_uninit()*.

A naive user might not follow this flow. Your program should handle such conditions gracefully.

4 Submission Instructions

1. There are no groups in this assignment. Everyone does and submits it individually.
2. You submission shoud consist of two .c files (one containing the main() and the other containign all your other functions), a .h file (containing the headers for your functions, and a make file.
3. You can name your submission as LecX_st0XXXX_A3_main.c and LecX_st0XXXX_A3_malloc.c:
4. You'll also submit a PDF file combining all the codes and the submitters' info. The name should follow the pattern LecX_st0XXXX_A3.pdf. A template will be provided later.
5. The PDF should list any and all resources you will have used to do this assignment i.e. web sites, books, people consulted. Copy pasting code from internet is not allowed.
6. The final submission should combine all these file together and submit one .tar.gz file e.g. LecX_st0XXXX_A3.tar.gz. Explore the linux 'tar' command to archive and zip multiple files/directories together.

5 Rubric

Category	Marks
Followed submission insns	10 marks
Code was readable + Compiled without warnings + Program handles errors well	10 marks
Each function working as intended with all the corner cases handled properly	10 marks each
Total	100 marks