Lab 6: Malloc lab

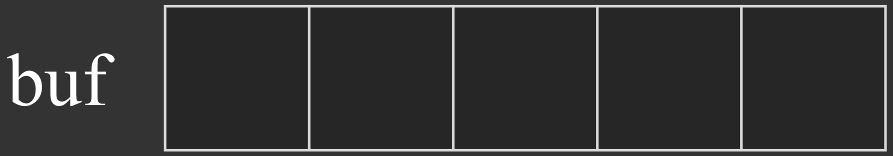
Due: 20181127

Contents

Dynamic memory allocation

Static	Dynamic

`#define LENGTH 5`
`void func1() {`
`    int buf[LENGTH];`
`}`
	`int len = 5;`
	`void func2() {`
	`    int *buf = (int *) malloc(len * sizeof(int));`
	`    free(buf);`
	`}`



Dynamic memory allocation

Static		Dynamic
By defining the variable	How to alloc	By calling malloc, calloc, realloc
Fixed at compile time	Length	Vary in running time
The program stack	Allocated in	The heap
Automatically	How to free	By calling free()

Objective

Design
a Dynamic Storage Allocator

correctly, efficiently, and fast

Specification

- Functions to implement in `mm.c`

```
int mm_init (void)
void *mm_malloc (size_t)
void *mm_free (void *)
void *mm_realloc (void *, size_t)
```

* Section 3 in the document will help you

Specification

In `memlib.c`,

```
    void *mem_sbrk (int incr)
    void *mem_heap_lo ()
    void *mem_heap_hi ()
    void *mem_heapsize ()
    void *mem_pagesize ()
    void mem_init()
    void mem_deinit()
    void mem_reset_brk()
```

Useful functions

Don't use.
These are for `mdriver`.

* Refer to section 4 in the document for more details

Specification

- You must not use standard allocation functions from `stdlib.h`
    - Using `malloc`, `calloc`, `realloc`, `free`, `sbrk`, `brk` is not allowed
- You must not define global/static compound data structures, but allowed to declare global scalar var
        - `int a[4]`: not allowed
        - `struct element b`: not allowed
        - `struct element *bp`: allowed
        - `int b`: allowed
    - change the interface of `mm.c`
- mem allocator must always return pointers aligned by 8-byte boundaries.
    - `0x40c30020` (o), `0x40c3001f` (x)

Checking

- Compile `mdriver`

```
$ make
```

- Run with default trace files

```
$ ./mdriver
```

- Specify a trace file: -f

```
$ ./mdriver -f traces/short1-bal.rep
```

- See the performance of `malloc()` in standard C library: -l

```
$ ./mdriver (-V) -l (-f <file>)
```

Grading policy

- Getting summary info from auto-grader

```
$ ./mdriver -g
Using default tracefiles in ./traces/
Perf index = 44 (util) + 9 (thru) = 53/100
correct:11
perfidx:53
```

- Maximum score mdriver gives
  - Correctness (11) – number of valid traces
  - Performance index (100) – equation written in document

Grading policy

- Your score = $\boldsymbol{correct} * \frac{\mathbf{20}}{\mathbf{11}} + \boldsymbol{perf} * \frac{\mathbf{35}}{\mathbf{100}}$
- Total 55 points
- There will be no style point.

Supplementary

- Consider both memory utilization and throughput for performance idx

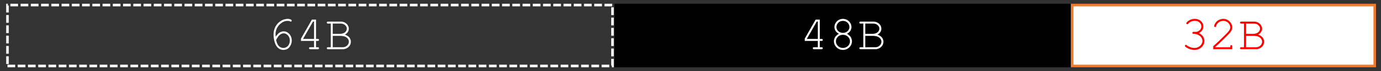
$$P = w\boxed{U} + (1 - w)\min\left(1, \frac{\boxed{T}}{T_{libc}}\right)$$

1. Throughput
   - # operations completed per second
2. Memory space utilization
   - Fragmentation managing

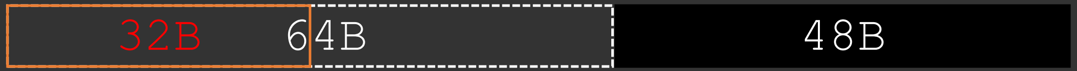
Supplementary

For an allocation sequence: `malloc(64B), malloc(48B), free(64B), malloc(32B)`

1. Allocate 32B next to 48B allocated memory
   - $U = \frac{48+32}{64+48+32} = 55\%$



2. Allocate 32B to memory that is freed before
   - $U = \frac{48+32}{64+48} = 71\%$



Precautions

- Check your code compiles successfully / does not crash the driver
  - You will receive zero point otherwise
- Not everything was covered on the slides
  - Please read the document in KLMS.

Preparing lab

- Fork repository to your account
  - https://gitlab-edu.kaist.ac.kr/CS230/lab6

- Clone repository

```
$ git clone ssh://git@gitlab-edu.kaist.ac.kr:10022/cs[your_student_id]/lab6.git
```

Handin

- Add mm.c to your git and commit

```
$ git add mm.c
$ git commit –m “Your commit message”
$ make handin
```

* Refer to the document.

Thank you