System Programming Project 4

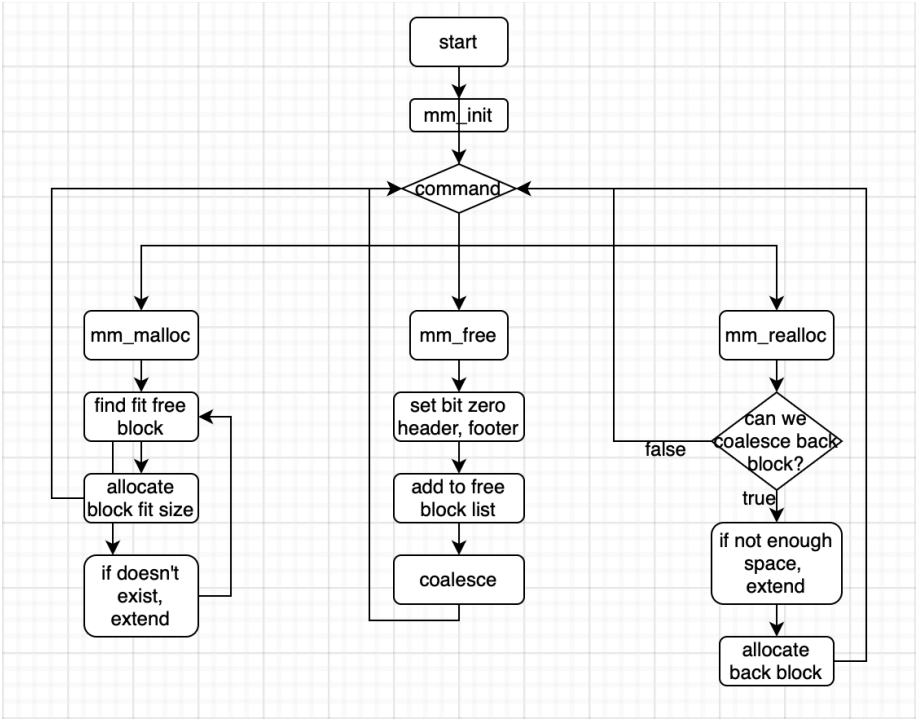
담당 교수 : 김영재

이름 : 윤준서

학번 : 20211558

1. Design

mm.c 의 flow char 는 다음과 같다.



프로그램을 시작하면 mm_init()을 통해 segregated list 설정을 한다. 프롤로그 블록과 에필로그 블록에 대한 초기 설정도 해당 함수에서 진행한다. 이후 명령어를 통해 malloc, realloc, free 세 가지의 기능을 실행한다.

mm_malloc()의 경우, 요구하는 크기와 가장 근접한 블록을 segregated list 에서 찾는다. 해당 크기의 블록이 존재하지 않는 경우 할당하려는 크기만큼 heap 을 extend()를 통해 늘려서 찾는다. 블록을 찾았다면 place()를 통해 할당을 완료하고 할당한 블록의 포인터를 반환한다.

mm_realloc()의 경우, 이미 할당된 블록의 크기가 요구하는 할당 크기 이상의 여유를 갖고 있다면 그래도 블록의 포인터를 반환한다. 더 할당이 필요한 경우, 앞뒤 블록과 합쳐서 다시 반환한다. 합칠 수 없는 경우, 새로운 블록을 malloc 을 통해 기존 블록의 내용을 복사하여 할당한다. 그리고 기존 블록은 free 한다.

mm_free()의 경우, header 와 footer 의 lsb 를 0 으로 설정하고, 블록을 segregated list 에 추가한다. 이때 앞뒤 블록과 합칠 수 있으면 합친다.

2. Description

Macro

#define ALIGNMENT : 블록이 메모리에서 맞춰지는 바이트 크기
#define WSIZE : word 크기
#define DSIZE : double word 크기
#define MAXSEGLIST : segregated list 의 크기
#define SMALLESTSIZE : 힙의 최소 크기
#define CHUNKSIZE : extend 를 실행 시 힙이 커지는 크기
#define MAX : 더 큰 값
#define MIN : 더 작은 값
#define PACK : 블록 크기와 할당 여부를 하나로 묶은 값
#define GET : 인자의 값을 4 바이트 만큼 읽은 값
#define PUT : 포인터 인자에 값을 4 바이트 만큼 작성
#define GET_SIZE : 블록 사이즈
#define GET_ALLOC : 블록의 할당 여부
#define HDRP : 블록의 헤더 포인터
#define FTRP : 블록의 푸터 포인터
#define NEXT_BLKP : 블록 포인터의 다음 포인터
#define PREV_BLKP : 블록 포인터의 이전 포인터

Global Variable

static void **seg : segregated list 자료구조 내부에 각 블록 크기에 따른 리스트들이 존재한다.

Function

static void insert(void* bp, size_t size)

- 주어진 블록 크기에 맞는 리스트의 인덱스를 찾는다.
- 해당 리스트가 비어 있으면 블록을 첫 번째 원소로 추가한다.
- 리스트가 비어 있지 않으면 블록을 리스트의 앞부분에 추가한다.

static void delete(void* bp)

- 해당 블록의 크기와 인덱스에 맞는 리스트를 찾아 블록을 삭제한다.
- 삭제할 블록이 리스트의 첫 번째 원소라면 리스트의 헤드를 업데이트한다.
- 첫 번째 원소가 아니라면 이전 또는 다음 포인터를 수정해 블록을 연결한다.

static void *coalesce(void *bp)

- 현재 블록과 이전 또는 다음 블록을 검사하여 병합할 수 있는 경우 병합하고, 새로 병합된 블록을 리스트에 다시 삽입한다.
- 병합된 블록의 크기를 수정하고, 병합 후에 다시 인접 블록과 병합을 시도한다.

static void *extend_heap(size_t words)

- 인자 크기만큼 새로운 메모리를 할당하고, 새로운 블록의 헤더와 푸터를 설정한다.
- 확장된 힙의 새로운 블록을 리스트에 추가한 후, 병합 여부를 확인한다.

int mm_init(void)

- segregated 리스트를 위한 메모리 공간을 할당하고, 내부의 각 리스트를 NULL 로 초기화한다.
- 프롤로그와 에필로그 블록을 설정하고, 힙을 확장하여 초기화한다.
- 힙 초기화가 완료되면 0 을, 실패하면 -1 을 반환한다.

void *find_space(size_t size)

- 각 리스트를 순차 탐색하면서 인자 크기와 맞는 블록을 찾는다.
- 찾은 블록을 반환하되, 찾지 못하면 NULL 을 반환한다.

static void *place(void *bp, size_t asize)

- 블록을 분할할 수 있으면 남은 블록을 처리하고, 새로 할당한 블록을 반환한다.
- 분할할 필요가 없으면 블록의 전체 크기를 할당된 상태로 처리하고, 남은 공간이 여유로울 경우 그대로 할당한다.

void *mm_malloc(size_t size)

- 요청한 크기만큼 메모리를 할당하고, 리스트에서 적합한 크기의 블록을 찾아 할당한다.
- 블록을 찾기 못하면 힙을 확장하여 추가로 메모리 블록을 할당한 뒤 다시 적합한 블록을 할당한다.

void mm_free(void *bp)

- 블록을 리스트에 삽입하고, 헤더와 푸터를 free 로 설정한다.
- 인접한 블록과 병합 여부를 확인하고, 병합 가능하면 병합한 블록을 반환한다.

void *mm_realloc(void *ptr, size_t size)

- ptr 이 NULL 이면 새로운 메모리 블록을 할당하고, 크기가 0 이면 해당 블록을 free 한다.
- 기존 블록의 크기보다 더 큰 크기를 요청하는 경우, 힙을 확장하거나 근접 블록과 병합한 블록을 새로 반환한다.
- 요청한 크기가 작은 경우 기존 블록을 그대로 사용하거나 새로 할당한 메모리에 데이터를 복사하여 반환한다.

Result

프로그램의 성능 결과는 다음과 같다.

```
cse20211558@cspro:~/SP/project4$ ./mdriver -v
[20211558]::NAME: Junseo Yun, Email Address: yjs2673@gmail.com
Using default tracefiles in ./tracefiles/
Measuring performance with gettimeofday().

Results for mm malloc:
trace  valid  util     ops      secs  Kops
 0       yes   97%    5694  0.000589  9662
 1       yes   97%    5848  0.000537 10894
 2       yes   98%    6648  0.000562 11840
 3       yes   98%    5380  0.000399 13494
 4       yes   99%   14400  0.000483 29838
 5       yes   92%    4800  0.001952  2459
 6       yes   90%    4800  0.001946  2467
 7       yes   60%   12000  0.027417   438
 8       yes   50%   24000  0.000672 35720
 9       yes  100%   14401  0.000272 53003
10       yes   87%   14401  0.000258 55905
Total          88%  112372  0.035085  3203

Perf index = 53 (util) + 40 (thru) = 93/100
```