Multicore Programming Project 3

담당 교수 : 최재승 교수님

이름 : 이승연

학번 : 20211569

1. 개발 목표

c언어로 dynamic storage allocator 만들기. 즉, 자신의 malloc, free, realloc 함수 작성하기.

2. design of allocator

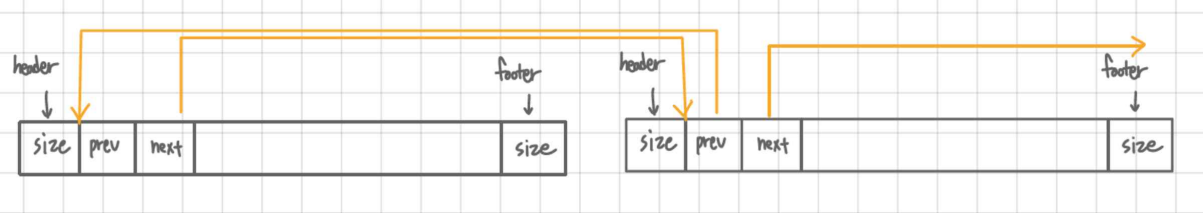
본 프로젝트에서는 implicit free list로 구현되어 있던 코드를 explicit free list로 변경하여 구현하였다. block 할당 시 전체 block이 아닌 free block만 탐색한다. 이를 위해 다음과 같이 block을 설계했다.

- Allocated block

각 block 내의 첫 word와 마지막 word에는 block size를 저장한다. 또한 현재 allocated 상태이기 때문에 least significant 3 bit에 1을 저장한다.

- Free block

각 block 내의 첫 word와 마지막 word는 allocated block과 같이 block size를 저장한다. 또한 현재 free 상태이기 때문에 least significant 3 bit에 0을 저장한다. header의 다음 word에는 이전 free block 주소를 가리키는 prev 포인터를 저장한다. 그 다음 word에는 다음 free block 주소를 가리키는 next 포인터를 저장한다.



코드에서 사용한 free_blk를 나타낸 그림이다. free_blk는 첫 번째 free block을 포인트한다. 첫 번째 free block의 prev에는 NULL값이 저장되어 있다.

3. 전역 변수 & 매크로 & 함수 설명

1) 전역 변수

explicit list로 dynamic storage allocator를 구현하는데 총 한 개의 전역변수를 이용했다.

static char *free_blk = NULL; 이 free_blk는 free block list의 첫 번째 block을 가리키는 포인터이다. LILO 방식을 사용했기 때문에 insert할 때는 항상 free_blk가 새롭게 추가할 block을 가리키도록 한다.

2) 매크로

```
34  /* single word (4) or double word (8) alignment */
35  #define ALIGNMENT 8
36
37  /* rounds up to the nearest multiple of ALIGNMENT */
38  #define ALIGN(size) (((size) + (ALIGNMENT-1)) & ~0x7)
39
40  #define WSIZE 4
41  #define DSIZE 8
42  #define CHUNKSIZE (1 << 12) // heap 확장 위한 기본 size
43
44  #define MAX(x, y) (x > y ? x : y)
45  #define GET(p) (*(unsigned int *)(p))
46  /* 크기, 할당 비트 통합해서 header, footer에 값 리턴 */
47  #define PACK(size, alloc) ((size) | (alloc))
48  /* 인자 p가 가리키는 워드에 val을 저장 */
49  #define PUT(p, val) (*(unsigned int *)(p) = (val))
50
51  #define HDRP(bp) ((char *)(bp) - WSIZE) // Header 포인터
52  #define FTRP(bp) ((char *)(bp) + GET_SIZE(HDRP(bp)) - DSIZE) // Footer 포인터
53
54  #define NEXT_BLKP(bp) ((char *)(bp) + GET_SIZE((char *)(bp) - WSIZE)) // 다음 block 포인터
55  #define PREV_BLKP(bp) ((char *)(bp) - GET_SIZE((char *)(bp) - DSIZE)) // 이전 block 포인터
56  #define SET_PTR(p, bp) (*(unsigned int *)(p) = (unsigned int)(bp))
57
58  #define GET_NEXT(bp) (*(char **)((char *)(bp) + WSIZE))
59  #define GET_PRED(bp) (*(char **)(bp))
60
61  #define GET_SIZE(p) (GET(p) & ~0x7) // 뒤에 3자리 없어짐
62  #define GET_ALLOC(p) (GET(p) & 0x1)
63
64  #define SIZE_T_SIZE (ALIGN(sizeof(size_t)))
65
```

교과서를 참고했고 실제 작성한 매크로는

GET_NEXT, GET_PRED이다. GET_NEXT는 다음 free block의 주소 GET_PRED는 이전 free block의 주소를 가리킨다.

3) 함수

- int mm_init(void)

최소 크기가 16바이트여야 하기 때문에 8워드 크기의 힙을 생성한다. free_blk는 free list의 첫 번째 block을 가리키는 포인터이다. 이 free_blk가 free_list의 첫 번째 block을 가리키도록 초기화한다.

- void *mm_malloc(size_t size)

allocate 요청이 들어오면 block의 크기는 최소 16바이트여야 하므로 size를 asize로 조정한다. find_fit 함수를 사용해 asize를 갖고 있는 block을 찾고 만약 찾았다면 place함수를 사용해 그 자리에 block을 할당한다. 만약 asize를 갖고 있는 block을 찾지 못했다면 extend_heap 함수를 사용해 힙을 확장시킨다.

- void mm_free(void *bp)

bp가 가리키는 block을 free한다. free한 후 만약 인접한 block들이 있다면 coalesce함수를 이용해 병합한다.

- void *mm_realloc(void *ptr, size_t size)

size만큼 block이 있는지 찾아 mm_malloc 함수를 호출해 할당하고,

static void place(void *bp, size_t asize)

asize block bp를 할당하는 함수이다. find_fit 함수에서 찾은 곳에 block을 할당하고 만약 자리가 남았다면 그 size만큼 다시 free block으로 만들고 free list에 추가한다.

- static void *find_fit(size_t asize)

free_blk부터 free list를 탐색하여 asize만큼의 자리가 남아있는 곳을 포인터로 반환한다. asize를 담을 수 있는 block이 있을 때까지 탐색한다.

- static void *extend_heap(size_t words)

힙을 요청받은 words만큼 확장한다. 힙 확장에 성공했다면 새 block의 header, footer을 초기화한다. 새 block과 이전 free list를 병합한 후 리턴된 block 포인터를 반환한다.

- static void *coalesce(void *bp)

block을 free 시키거나 allocate시킬 때 만약 인접해 있는 block들이 있다면 두 block을 합쳐 하나

의 block으로 만들어 준다. 총 4가지의 경우의 수를 생각했다. 1) 앞, 뒤 block 빈 경우 2) 앞 block만 빈 경우 3) 뒤 block만 빈 경우 4) 앞 뒤 block 모두 할당된 경우.

- static void delete_freeblkp(void *bp)

bp를 free list에서 제거하는 함수이다. 이전 free block을 다음 free block으로 연결하고, 다음 free block의 prev를 이전 free block으로 연결시켜서 bp를 제거한다.

- static void insert_freeblkp(void *bp)

LIFO(Last in First out)방식을 사용해서 bp를 free list에 삽입할 때 list의 첫부분에 삽입하도록 구현했다.

4. 출력 결과

```
cse20211569@cspro:~/prj3-malloc$ ./mdriver
[20211569]::NAME: Seungyeon Lee, Email Address: tmddus4671@sogang.ac.kr
Using default tracefiles in ./tracefiles/
Perf index = 45 (util) + 40 (thru) = 85/100
```

mdriver을 실행했을 때 performance는 85가 나오는 것을 확인할 수 있었다.