System Programming Project 3

담당 교수 : 김영재

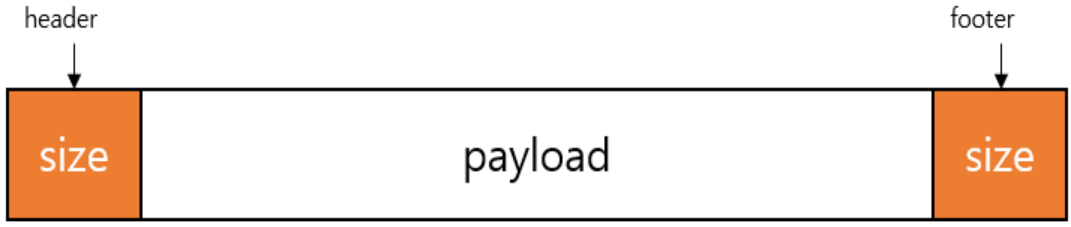
이름 : 20181654

학번 : 오상훈

1. **Design of allocator**

본 프로젝트에서 dynamic memory allocator 를 구현하기 위해서 explicit free list 를 적용하여 구현하였다. Explicit free list는 heap memory 영역에서 free block들만을 리스트를 유지하여 메모리를 관리한다. 이를 위해서 allocated block과 free block의 구조를 다음과 같이 설계하였다.

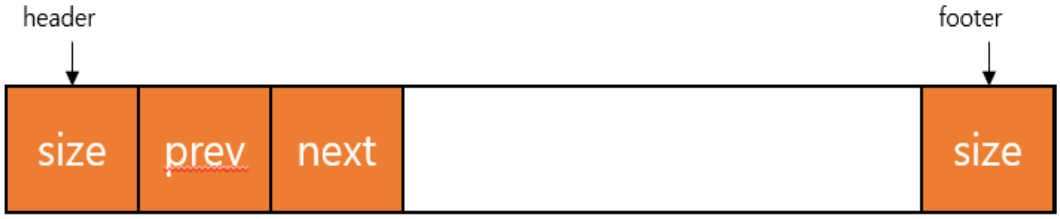
- Allocated block



Allocated block에서는 첫 번째 word에 block의 크기를 저장한다. 마지막 word에는 coalescing을 위해 boundary tag로 header와 같이 block의 크기를 저장한다.

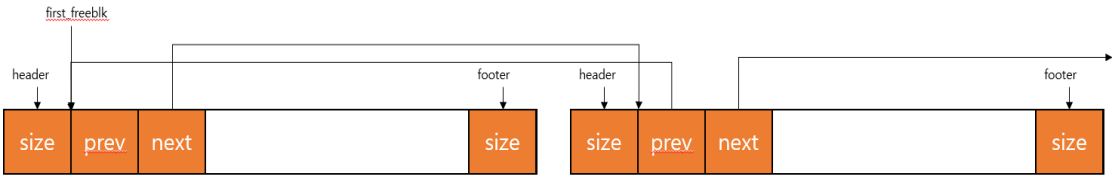
이때, block은 double word alignment 를 하므로 least significant 3 bit가 항상 0이게 된다. 따라서 least significant bit에 allocate bit로 1을 저장한다. 또한, payload의 크기가 block의 크기보다 작으므로 internal fragmentation 이 존재하게 된다.

- Free block



Free block에서는 header와 마찬가지고 첫 번째 word와 마지막 word에 block의 크기를 저장한다. Header의 다음 word에는 이전 free block의 주소 값을 저장하고, 그 다음 word에는 다음 free block의 주소 값을 저장하여, explicit free list를 유지한다. Allocated block에서와 마찬가지로 header와 footer의 마지막 3bit는 사용되지 않기 때문에, 마지막 bit에 0을 저장한다.

Explicit free list는 free block의 prev와 next 포인터로 연결하므로 다음과 같은 구조를 가진다.



first_freeblk는 첫 번째 free block을 포인팅하고 있다. 첫 번째 free block의 prev에는 null값이 저장되어있으며, 그 다음 free block을 next 포인터를 통해 연결한다. Explicit free list는 리스트에 노드 insertion, deletion 작업의 시간복잡도를 O(1)로 하기 위해 LIFO(last-in-first-out) 형태로 구성하였다.

2. Description of subroutines, global variables

2-1. global variables

Dynamic memory allocator 를 explicit list로 구현하는데에 사용된 전역변수는 총 2개이다.

먼저 heap_listp는 heap에 첫 번째 block을 가리키는 특수한 block이다. Header와 footer로만 이루어져있다.

나머지 전역변수 하나는 First_freeblk으로 free block list의 첫 번째 free block을 가리키고 있다.

2-2. subroutines

- mm_malloc

  mm_malloc함수는 메로리를 동적할당해주는 함수이다. 할당하고자 하는 블록의 size를 입력받는데, size가 0이라면 NULL을 return한다. Find_fit함수를 호출하여, free list에 있는 free block들 중, 할당할 수 있는 블록의 주소를 return받는다. 만약 할당할 수 있는 free block이 없을 경우, extend_heap함수를 호출하여 heap의 크기를 늘리고, place함수를 호출하여 블록을 할당한다.

- mm_free

  mm_free함수는 동적할 당한 메모리를 deallocate하고자 할 때 호출한다. 블록의 header와 footer의 allocate bit를 0으로 하여 free block으로 만든다. coalesce함수를 호출하여, false fragmentation이 생기지 않도록 이전 블록과 다음 블록과 합친다.

- mm_realloc

mm_realloc함수는 이전에 malloc 또는 realloc함수 호출을 통해 할당 받은 블록의 크기를 update하고자할 때 호출하는 함수이다. 재할당하고자하는 블록의 주소와 크기를 입력받는데, 블록 포인터 ptr이 NULL이라면 mm_malloc을 수행하여 return 하고, size가 0이라면 mm_free를 수행하여 return 한다. 만약 새로 update하려는 블록의 크기가 기존 크기보다 작을 경우, 다른 작업을 수행하지 않고, return 한다. 다음 블록이 free block이고, 현재 블록과 다음 블록을 합친 크기가, 새로 할당하려는 크기보다 크다면, 기존 블록을 mm_free하고, mm_malloc을 호출하는 불필요한 작업을 할 필요 없으므로, 다음 블록을 free list에서 삭제하고, 블록의 크기를 update한다.

위의 경우가 모두 아니라면, 새로운 블록을 mm_malloc을 통해 할당 받고, payload를 복사한 뒤, 기존 블록을 free 시킨다.

- insert_freeblk

  insert_freeblk은 free list에 블록을 추가할 때 호출하는 함수이다. insert하려는 블록을 next_pointer에 기존 first_freeblk의 주소를 저장하고, 기존 first_freeblk의 prev_pointer에 insert 하려는 블록의 주소를 저장한다. 또한, insert하는 블록의 prev_pointer는 NULL 로 set하고, first_freeblk 전역변수를 insert block으로 update한다.

- remove_freeblk

  remove_freeblk은 free list에서 블록을 제거할 때 호출하는 함수이다. 만약 제거하고자 하는 블록에서 prev_pointer가 null이라면 첫 번째 블록이었음을 의미하므로, first_freeblk 포인터이 다음 블록을 가리키도록 한다. 새로운 first_freeblk의 prev_pointer에 null을 저장한다.

- coalesce

  coalesce함수는 free block하고자 하는 블록의 이전 블록 또는 다음 블록이 어떤 블록인지에 따라 다른 과정을 수행한다.

  Case1(이전, 다음 블록이 모두 allocated 블록인 경우) : insert_freeblk함수를 호출하여 free list에 free 하고자 하는 블록을 insert한다.

  Case2(이전 블록은 allocated, 다음 블록은 free block인 경우) : 현재 block 포인터 변수 bp에 해당하는 블록을 free list 에서 제거한다(remove_freeblk). 현재 블록을 다음 블록과 합쳐야하므로, bp의 원래 header와 다음 블록의 footer에 두 블록을 합친 크기를 저장하고, free list에 합친 블록을 insert한다.

  Case3(이전 블록은 free, 다음 블록은 allocated block인 경우) : 이전 블록을 free list에서 제거한다(remove_freeblk). 이전 블록의 header와 현재 블록의 footer에 두 블록

을 합친 크기를 저장하고 bp가 이전 블록을 point하도록 한다. 합친 블록을 free list에 insert한다.

Case4(이전, 다음 블록이 모두 free block인 경우): 이전 블록과 다음 블록, 현재 블록을 하나의 블록으로 합쳐야 하므로, 이전 블록의 header와 다음 블록의 footer에 세 블록을 합친 크기를 저장한다. 이전 블록과 다음 블록을 free list에서 제거한 뒤, 합친 블록을 free list에 insert 한다.

- extend_heap

  extend_heap함수는 heap에 할당할 수 있는 block이 없을 때, sbrk함수를 통해 heap의 크기를 증가시키는 함수이다. 증가시킬 heap의 word size를 입력 받아 이를 double word alignment 한 크기를 구한다. memlib.c의 mem_sbrk함수를 호출하여 heap의 크기를 늘리고, 다음과 같이 증가시킨 heap의 첫 번째 블록의 포인터 bp 의 header 와 footer에 크기를 저장하고, epilogue header를 할당한다.

```
/* Initialize free block header/footer and the epilogue header */
PUT(HDRP(bp), PACK(size, 0));
PUT(FTRP(bp), PACK(size, 0));
PUT(HDRP(NEXT_BLKP(bp)), PACK(0, 1));
```

- find_fit

  find_fit함수는 first_freelk 부터 시작하여 입력 받은 크기를 할당할 수 있는 free block을 찾아서 return 한다.

- place

  place함수는 할당할 free 블록의 주소와 할당할 블록의 크기를 입력받는다. 기존 freeblock에 할당하므로, remove_freeblk함수를 호출하여, free list에서 block을 remove 한다. 기존 free block의 header와 footer에 할당할 블록의 크기와 allocate bit을 같이 저장한다. 이때, 블록이 split되는 경우, split한 이후의 다음 블록은 free block이므로, freeblock의 크기를 header와 footer에 저장하고, free list에 insert한다.

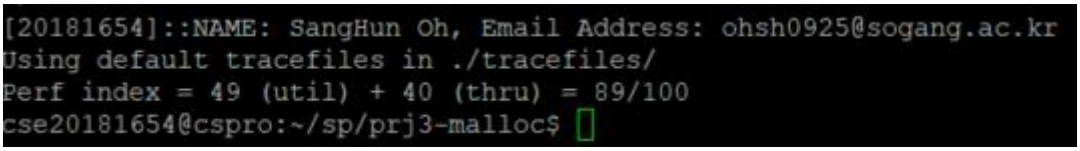
3. mm_check

   mm_check함수를 통하여, 구현 과정에서 heap consistency를 확인하였다. heap consistency를 확인하기 위해서 다음과 같은 조건들을 만족하는지 검사하는 코드를 작성하여 검사하였다.

   3-1. 블록이 double word aligned 되어있는지 확인

   3-2. 블록의 header에 저장되어 있는 size, allocate bit가 footer에 저장되어 있는 size, allocated bit와 같은지 확인

4. 결과



구현한 dynamic memory allocator를 검사하기 위해 mdriver를 실행하였을 때, 위의 사진과 같이 performance index는 89가 측정되는 것을 확인할 수 있었다.

여러 client와의 통신을 동시에 처리하는 서버를 구현하였다.