Multicore Programming Project 3

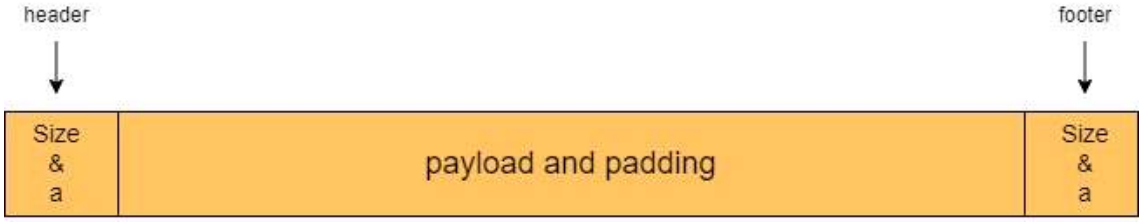
담당 교수 : 박성용 교수님
이름 : 이용욱
학번 : 20191626

1. 개발 목표

Dynamic Memory Allocator를 직접 만드는 것이 이번 프로젝트의 목표이다. 기존에 c언어 헤더파일에 내장된 malloc , realloc, free 등의 기능을 하는 함수를 만드는 것이다. Free blocks를 관리하는 방법에 따라 implicit , explicit , segregated list 등 여러 가지 방법이 있는데 이 중 적합한 방법을 선택해서 개발한다.

2. Design of allocator

이번 프로젝트에서 Dynamic Memory Allocator를 구현하기 위해서 implicit free list를 적용해서 구현 하였다.



위는 block의 구조를 표현한 것이다. allocated blockdpgtjsms 첫 번째 word에 block의 크기를 저장한다. 그리고 마지막 word에는 coalesce를 위해서 header와 같이 size를 저장한다.

이때 block은 8byte 단위로 alignment 하므로 lsb 3bit이 항상 0이다.

따라서 이를 활용하기 위해 allocate bit로 0 또는 1을 저장하여 free block 과 allocated block을 표현한다.

그리고 implicit free list방식으로 block을 포인터로 연결한다.

여기서 next_fit 방식을 활용해서 가용 블록을 찾는 방식을 선택하였다.

3. 함수, 전역 변수 설명

3.1

- 전역 변수로는 static *heap_listp와 static *temp_bp를 사용한다.

3.2

- int mm_init(void)

mm_init 함수를 호출함으로써 heap을 초기화 해준다.

memory system에서 4* wordsize 만큼 받은 후에 empty free list를 만들어서 초기화 한다.

그 이후 extend_heap함수를 불러 heap을 CHUNKSIZE만큼 확장하고 , 초기 free 블록을 만든다. 이 후 allocator는 초기화 되고, allocated or free 상태가 된다.

- void *mm_malloc(size_t size)

2 word 이하의 사이즈는 4word로 할당 요청하고, 2words를 초과할 시에는 충분한 양의 8Byte 배수의 용량을 할당한다.

그리고 find_fit 함수를 호출해서 적당한 크기의 가용 블록을 검색한다. 그리고 place 함수로 초과 부분을 분할하고, 새롭게 할당한 블록의 포인터를 반환한다.

만약 가용블록이 찾아지지 않는다면 memory를 추가 할당하고 다시 place 함수를 호출한다.

- void * mm_free(void *bp)

bp 가 가리키는 block pointer 영역의 메모리를 해제한다.

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

bp가 NULL인 경우 mm_malloc을 , heap_listp 가 0인 경우 mm_init을 호출한다.

이 때 새로 할당하는 new size가 기존 size보다 큰 경우에는 다음 가용 블록의 크기를 고려하여 새롭게 블록을 할당하거나 이전 블록을 그대로 사용한다. 새롭게 블록을 할당하는 경우에는 memcpy를 활용하여 이전 메모리를 복사하여 할당한다.

- static void *extend_heap(size_t words)

교재 에 있는 함수를 활용하였다. 간략하게 설명하자면 초기화된 heap을 확장하는 함수이다. words 에 따라 확장하는 heap의 크기가 달라진다.

- static void *coalesce(void *bp)

coalesce 함수는 free block 하고자 하는 블록의 prev block 또는 next block이 free 상태인지 allocated 상태 인지에 따라 다른 과정이 수행된다.

교재에 4가지의 경우가 소개되어 있다. 나는 다음의 case에 따라 coalesce를 구현하였다.

case1 : Prev && Next allocated

temp_bp 에 bp 주소를 담고, bp를 return한다.

case2 : Prev allocated, Next free

size 에 next 블록의 사이즈를 추가하고, 그 크기만큼 bp에 새로이 할당한다.

case3 : Prev free, Next allocated

size에 free 블록의 사이즈를 추가하고, 크 크기만큼 bp에 새로이 할당한다. 그리고 bp가 prev block을 가리키도록 한다.

case4 : Prev free, Next free

이전 블록과 다음 블록, 현재 블록을 모두 하나의 블록으로 합쳐야 하므로, 그 크기만큼을 size에 추가해 준다. 그리고 그 크기만큼 새롭게 할당한다. 그리고 bp가 prev block을 가리키도록 한다.

- static void *place(void * bp, size_t size)

place 함수는 할당할 free block의 주소와 asize를 입력받는다. 현재 용량과 asize의 용량 차이가 16 byte보다 크거나 같다면 요청한 용량만큼 블록을 배치한다. 그리고 bp에 다음 블록의 주소를 할당한다. 남은 블록에는 header , footer를 배치한다.

현재 용량과 asize 차이가 16byte보다 작다면 해당 블록을 그대로 사용한다.

static void *find_fit(size_t asize)

find_fit 함수는 입력받은 크기를 할당 할 수 있는 free 블록을 찾아서 return 한다.

여기서 find_fit 의 구현을 나는 next_fit 방식으로 하였는데, temp_bp에 담아둔 마지막 할당 위치를 활용하여 가용 블록을 찾아서 리턴하도록 한다.

4. 구현 결과

./mdriver -V를 실행하여 다음과 같은 결과를 얻을 수 있다.

```
Measuring performance with gettimeofday().

Testing mm malloc
Reading tracefile: amptjp-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: cccp-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: cp-decl-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: expr-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: coalescing-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: random-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: random2-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: binary-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: binary2-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: realloc-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.
Reading tracefile: realloc2-bal.rep
Checking mm_malloc for correctness, efficiency, and performance.

Results for mm malloc:
trace  valid  util     ops      secs  Kops
 0       yes   95%    5694  0.006341   898
 1       yes   94%    5848  0.002656  2202
 2       yes   97%    6648  0.006852   970
 3       yes   98%    5380  0.006405   840
 4       yes   66%   14400  0.000192 75117
 5       yes   93%    4800  0.007034   682
 6       yes   90%    4800  0.006275   765
 7       yes   55%   12000  0.022523   533
 8       yes   51%   24000  0.009681  2479
 9       yes   80%   14401  0.000262 55008
10       yes   46%   14401  0.000121119411
Total          79%  112372  0.068341  1644

Perf index = 47 (util) + 40 (thru) = 87/100
cse20191626@cspro:~/cse4100/project3/prj3-malloc$
```