Multicore Programming Project 3

담당 교수 : 박성용

이름 : 김승원

학번 : 20182186

1. 개발 목표

C 프로그램을 위한 동적 메모리 할당기인 malloc, free, realloc 함수의 사용자 정의 버전을 작성한다. 창의적으로 설계 공간을 탐구하고, 효율적이고 빠른 할당을 할 수 있도록 구현하는 것을 목표로 한다.

2. 개발 범위 및 내용

A. 개발 범위

1) Memory allocation Functions

- mm_Malloc()

- mm_free()

- mm_realloc()

2) Memory Management Functions

- extend_heap()

- coalesce()

3) Helper Functions

- mm_init()

- place()

- insert_node()

- delete_node()

B. 개발 내용

C 프로그램을 위한 동적 메모리 할당과 관련한 함수를 개발한다. 해당 allocator의 핵심 기능인 malloc, free, realloc 함수를 직접 구현하고, 이를 지원하기 위해 필요한 추가적인 함수들을 구현한다.

Mm_malloc은 exp_listp 포인터를 사용해서 리스트를 순회하도록하고, 충분한 크기의 블록을 찾아 메모리를 할당한다. Best_fit 방식을 사용한다. 블록이 요청된 사이즈보다 크거나 같고 현재까지 찾은 블록보다 작을 때마다 최적의 블록을 업데이트한다. 최적의 블록 사이즈와 요청된 사이즈가 정확히 일치하는 경우, 그 즉

시 검색을 중지하고 해당 블록을 할당하도록 한다. 리스트를 모두 순회한 후에도 적합한 블록을 찾지 못하면, 새로운 힙 메모리를 확장하도록 구현한다.

Mm_free는 동적 메모리 할당을 통해 얻은 메모리 블록을 해제한다. Mm_realloc은 이미 할당된 메모리 블록의 크기를 재조정하는 역할을 한다. 위의 모든 함수들은 메모리 관리 기법 중 하나인 explicit free lists 방식을 이용하고, 이 방식은 해제된 블록을 연결 리스트 형태로 관리해서 메모리를 효율적으로 관리한다. 또한 이 방식을 메모리 블록을 최적의 위치에서 찾고, 사용 후에는 가능하면 합치는 방법으로 메모리를 효율적으로 활용하도록 한다.

C. 개발 방법

1) MACRO와 Global variable

ALIGNMENT: 워드 크기 8바이트를 정의

WSIZE: Word와 header/footer의 크기를 4바이트로 정의

DSIZE: Doubleword의 크기를 8바이트로 정의

CHUNKSIZE: 힙이 확장될 때의 바이트 단위 크기를 정의

INITCHUNKSIZE: 초기 heap 크기를 정의

MIN_BLOCK_SIZE: 최소 블록 크기를 16바이트로 정의

MAX, MIN: 두 개의 값 중에서 최대값과 최소값을 반환하는 매크로

ALIGN: 주어진 사이즈를 가장 가까운 ALIGNMENT의 배수로 올림하는 매크로

SIZE_T_SIZE: size_t의 크기를 ALIGNMENT의 배수로 올림하는 매크로

PACK: 주어진 사이즈와 할당 비트를 하나의 word로 묶는 매크로

GET, PUT: 주어진 주소에서 word를 읽고 쓰는 매크로

GET_SIZE, GET_ALLOC: 주어진 주소에서 사이즈와 할당 비트를 읽는 매크로

HDRP, FTRP: 주어진 블록 포인터에서 header와 footer의 주소를 계산하는 매크로

NEXT_BLKP, PREV_BLKP: 주어진 블록 포인터에서 다음 블록과 이전 블록의 주소를 계산하는 매크로

TARGET_PTR: 주어진 포인터 'p'의 값을 'ptr'로 설정하는 매크로

GET_NEXT_PTR, GET_PREV_PTR: 주어진 free block pointer에서 다음, 이전 포인터의 주소를 얻는 매크로

SEGLIST_NEXT, SEGLIST_PREV: segregated free list에서 주어진 free block pointer의 다음, 이전 free block pointer를 얻는 매크로

BUFFER_SIZE_4_REALLOC: realloc 요청을 위한 버퍼 크기를 정의

exp_listp: free list의 시작을 가리키는 전역 포인터

2) Memory Allocation Functions:

- mm_malloc(): 매개변수 size를 검사해서 0인 경우 Null을 반환하도록 한다. 할당할 메모리 블록의 크기를 조정해서 adj_block_size에 저장하고, 적합한 메모리 블록을 가리키는 best_fit 변수와 해당 블록의 크기를 저장하는 best_fit_size변수를 초기화한다. 이후 연결 리스트에서 메모리 블록을 탐색하여 가장 적합한 블록을 찾고, 적합한 블록이 없는 경우에 heap을 확장해서 메모리를 할당한다. Best_fit 블록에 adj_block_size 크기의 메모리를 할당하고, 해당 블록의 포인터를 반환한다.

- mm_free(): ptr(size)의 크기를 구하고, get_size 함수와 HDRP 매크로를 사용해서 해당 블록의 헤더에서 크기 정보를 읽어온다. ptr이 가리키는 블록을 더 이상 할당되지 않은 상태로 표시하기 위해 pack 함수를 사용해서 크기와 할당 여부를 packging하고, 이 정보를 HDRP와 FTRP 매크로를 이용해서 블록의 header와 footer에 기록한다. 해제된 블록을 free list에 추가하고, 이 때 insert_node 함수를 사용해서 블록과 그 크기를 인자로 전달한다. 이후 coalesce를 이용해서 해체된 블록 주변에 또 다른 해제된 블록이 있는지 확인하고, 있다면 이들을 합친 다음 더 큰 해제된 블록을 만들어준다.

- mm_realloc(): 요청된 size를 적절하게 조정한다. Size가 DSIZ보다 작거나 같으면 2*DSIZE로 설정하고 그렇지 않은 경우 size + DSIZE를 8바이트 경계로 정렬한다. 현재 블록의 크기(cur_size)를 구하고, 이 크기가 조정된 adj_size보다 크거나 같은지 확인한다. 만약 그렇다면, 현재 블록을 재사용하되 필요한 경우 블록을 분할한다. 현재 블록의 크기가 조정된 size보다 작은 경우, 다음 블록이 해제된 상태

이고, 그 크기가 필요한 크기를 충족하는지 확인한다. 만약 충족한 경우 현재 블록과 다음 블록을 합친다. 만약 모두 만족하지 않으면 새로운 블록을 할당하고, 데이터를 복사한 다음 원래 블록을 해제시킨다.

3) Memory Management Functions:

- extend_heap(): 이 함수는 힙의 크기를 확장하는 역할을 한다. 이 함수는 지정된 단어 수(words) 만큼의 새로운 공간을 할당 받고, 이를 활용하여 새로운 free block를 초기화한다. 새로 생성된 free block는 free block list에 추가한다. 만약 이전 블록이 free 상태라면 새로운 블록과 합친다.

- coalesce(): 이 함수는 인접한 free block들을 합쳐 큰 하나의 free block으로 만드는 역할을 한다. 이 함수는 주어진 블록(bp)의 이전 블록과 다음 블록이 free 상태인지를 확인하고, free 상태라면 이들을 하나로 합치도록 한다. 합쳐진 큰 블록은 free block list에 다시 추가된다.

4) Helper Functions:

- mm_init(): 이 함수는 메모리 관리를 위한 초기 설정을 수행한다. 힙을 초기화하고 프롤로그(prologue)와 에필로그(epilogue)를 설정한다. 프롤로그와 에필로그는 블록 경계를 표시하는 역할을 하며, 힙 영역의 시작과 끝에 위치한다. 초기화 이후에는 extend_heap 함수를 호출하여 힙을 초기 크기(INITCHUNKSIZE)만큼 확장하고, 초기화가 성공하면 0을 반환하고, 실패하면 -1을 반환한다.

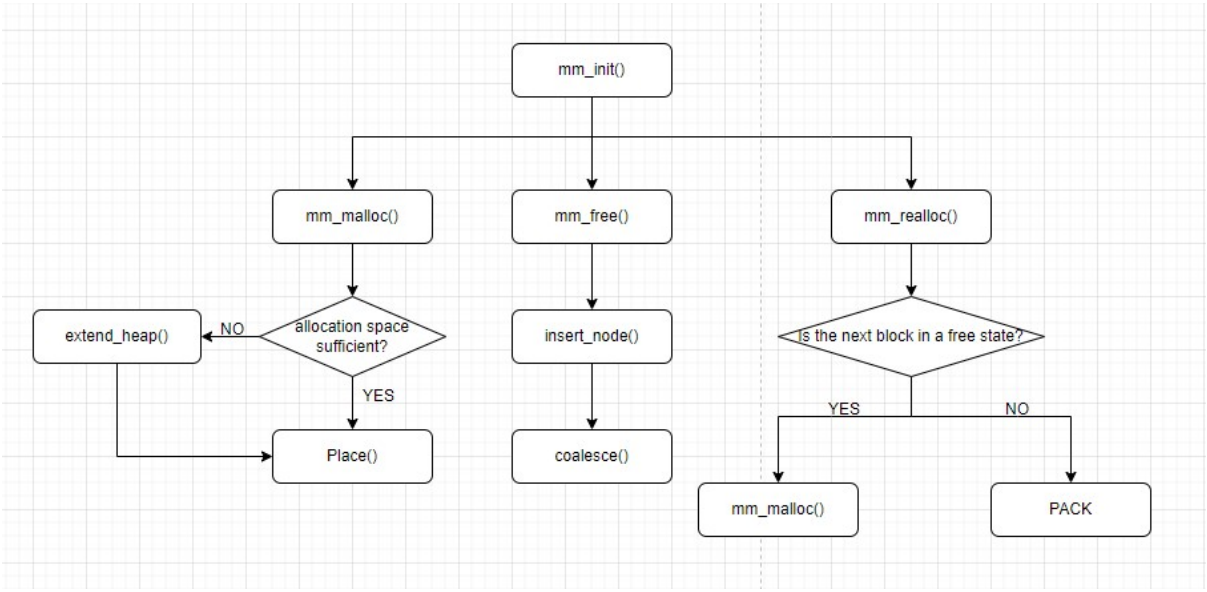
- place(): 이 함수는 주어진 포인터(bp)와 블록 크기(block_size)를 사용하여 메모리 블록을 할당한다. 현재 블록(bp)의 크기(cur_size)와 요청된 블록 크기(block_size)를 비교하여 분할이 가능하고, 남은 크기(remain_size)가 충분하다면 블록을 분할한다. 분할 후 남은 블록은 다시 free list에 삽입된다.

- insert_node(): 이 함수는 주어진 포인터(ptr)와 크기(size)를 사용하여 새 노드를 free list에 삽입합니다. 이 함수는 free list를 크기에 따라 정렬하여 관리하며, 새 노드는 그 크기에 따라 적절한 위치에 삽입된다.

- delete_node(): 이 함수는 주어진 포인터(ptr)가 가리키는 노드를 free list에서 제거한다. 이 함수는 노드가 free list의 시작, 중간, 끝에 있는 경우를 각각 처리한다. 해당 노드를 free list에서 제거한 후, 해당 노드의 이전 노드와 다음 노드를 서로 연결한다.

3. 구현 결과

A. Flow Chart



B. 구현결과 테스트 및 성능 평가

```
cse20182186@cspro:~/sp/prj3/prj3-malloc$ ./mdriver -V
[20182186]::NAME: Seungwon Kim, Email Address: spraucy76@gmail.com
Using default tracefiles in ./tracefiles/
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   99%    5694  0.000720  7909
 1       yes   99%    5848  0.000450 13007
 2       yes   99%    6648  0.000950  7001
 3       yes   99%    5380  0.000535 10060
 4       yes   99%   14400  0.000273 52728
 5       yes   95%    4800  0.012289   391
 6       yes   95%    4800  0.011294   425
 7       yes   95%   12000  0.025485   471
 8       yes   88%   24000  0.028745   835
 9       yes   87%   14401  0.000298 48277
10       yes   85%   14401  0.000182 79257
Total          95%  112372  0.081220  1384

Perf index = 57 (util) + 40 (thru) = 97/100
```